The University of Manchester

MATH30000 - Undergraduate project

The Mathematics of Computerised Tomography

An introduction to image reconstruction

Yngve Mardal Moe
Student Number: 9858213

supervised by
Dr. Sean Holman

July 24, 2016

Contents

1 Introduction

1.1 About this report

I want start by thanking my supervisor, Dr. Sean Holman, for his excellent supervision and support throughout the work of this report.

The problem of computerized tomography (CT) consists of reconstructing an image, described as a function of two variables, from a set of integrals over lines. CT has applications ranging from medical to industrial imaging and the theoretical background behind it led to Hounsfield and Cromack's nobel prize in 1979[7]. An interesting note is that the theoretical framework was in fact laid out by the famous mathematician Johann Radon in 1917[15], but it was not until its rediscovery in the 1970's that it got much attention. That is because reconstructing an image is computationally heavy, and there were no feasible ways accomplish this when Radon published his paper.

The CT-image reconstruction problem is a so-called inverse problem, a category of problems that is based around fitting parameters or initial conditions for a model to observed data. It is thus a quite advanced branch of mathematics, which requires a mathematical tools ranging from functional analysis to numerical optimisation. Inverse problems arise in many different fields, ranging from quantum physics to medical imaging and this text will give an introduction to its theoretical basis, while focusing on the CT-image reconstruction problem.

Getting a firm understanding of inverse problems is important in order to gain a good understanding of image reconstruction. This text will for that reason make the reader familiar with the theory of inverse problems and how they relate to CT-scans. This requires some prerequisites and it is assumed that the reader is familiar with some fourier analysis, numerical analysis and has a firm understanding of linear algebra. A noise model, not equal to white noise, will later be introduced and tested in order to best conduct numerical experiments. The last few chapters will introduce and discuss all the numerical tools used in CT-image reconstruction before conducting numerical experiments, simulating and comparing the different reconstruction methods. The simulations and comparisons were all conducted with a 128×128 pixel Shepp-Logan phantom as seen in figure 1.1. It is also unfortunately the case that the printed version of this report doesn't have as clear contrasts as the digital version, so many figures are much more informative on screen than paper.

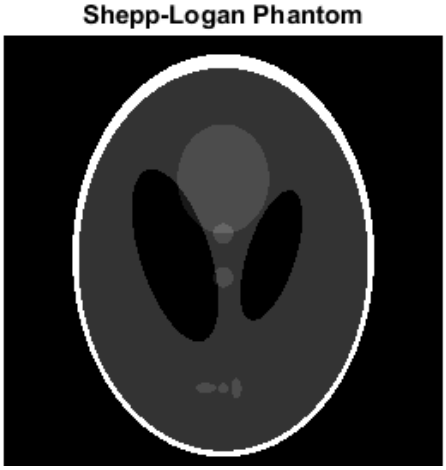

Figure 1.1: The phantom used in this text

1.2 Notation and prerequisites

There is a lot of different notation in use throughout the mathematical literature so I will, in order to avoid confusion, define a couple of important concepts here.

Definition 1.1 (n-dimensional Fourier transform)**.** Given a function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{C}$ with $\int_{\mathbb{R}^n} |f(\boldsymbol{x})|^2 d\boldsymbol{x} < \infty$. Its **Fourier transform** $\mathscr{F}_x f(\boldsymbol{\sigma})$ is given by

$$\mathscr{F}_x f(\boldsymbol{\sigma}) = \hat{f} = (2\pi)^{-n/2} \int_{\mathbb{R}^n} e^{-i\boldsymbol{\sigma}\cdot\boldsymbol{x}} f(\boldsymbol{x}) d\boldsymbol{x}. \tag{1.1}$$

We write $\mathscr{F}_x$ in order to denote what variable(s) to transform, as it is possible to transform k-variables of an n-dimensional function. An example of such a transform is $f(\boldsymbol{y}, \boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}$ with $\boldsymbol{y} \in \mathbb{R}^k$ and $\boldsymbol{z} \in \mathbb{R}^l$ and $k + l = n$.

$$\mathscr{F}_y f(\boldsymbol{\sigma}, \boldsymbol{z}) = (2\pi)^{-k/2} \int_{\mathbb{R}^k} e^{-i\boldsymbol{\sigma}\cdot\boldsymbol{y}} f(\boldsymbol{y}, \boldsymbol{z}) d\boldsymbol{y}. \tag{1.2}$$

The inverse Fourier transform $\mathscr{F}_x^{-1}$ is defined as

$$\mathscr{F}_x^{-1} f(\boldsymbol{x}) = (2\pi)^{-n/2} \int_{\mathbb{R}^n} e^{i\boldsymbol{\sigma}\cdot\boldsymbol{x}} \mathscr{F}_x f(\boldsymbol{\sigma}) d\boldsymbol{\sigma}. \tag{1.3}$$

Definition 1.2. Let $f, g : \mathbb{R}^n \to \mathbb{C}$, with $\int_{\mathbb{R}^n} |f(\boldsymbol{x})|^2 d\boldsymbol{x} < \infty$ and $\int_{\mathbb{R}^n} |g(\boldsymbol{x})|^2 d\boldsymbol{x} < \infty$. The **convolution** of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is defined as

$$f * g(\boldsymbol{t}) = \int_{\mathbb{R}^n} f(\boldsymbol{x}) g(\boldsymbol{t} - \boldsymbol{x}) d\boldsymbol{x}. \tag{1.4}$$

With $t \in \mathbb{R}^n$. We say $f * g$ as f convolved with g. We can also take the convolution of k variables over an n dimensional function, $f(\boldsymbol{x}, \boldsymbol{y})$ with $\boldsymbol{x} \in \mathbb{R}^k$, $\boldsymbol{y} \in \mathbb{R}^l$ and $k + l = n$, this is denoted

$$f *_x g(\boldsymbol{t}, \boldsymbol{y}) = \int_{\mathbb{R}^k} f(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{t} - \boldsymbol{x}) d\boldsymbol{x}. \tag{1.5}$$

The next theorem will be very important for understanding one of the CT-reconstruction algorithms.

Theorem 1.1 (Convolution theorem)
Let $f(\boldsymbol{x}), g(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. The fourier transform of the convolution $f * g(\boldsymbol{t})$, $\mathscr{F}_t f * g(\boldsymbol{\sigma})$ is given by the following product:

$$\mathscr{F}_t f * g(\boldsymbol{\sigma}) = (\mathscr{F}_x f)(\boldsymbol{\sigma})(\mathscr{F}_x g)(\boldsymbol{\sigma}). \tag{1.6}$$

It is also easy to use this theorem to prove that convolution is commutative. The next important definition is the notion of compact support.

Definition 1.3. We say that a function f has **compact support** if there exists an $r \in \mathbb{R}$ s.t. $f(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ with $||\boldsymbol{x}|| \geq r$

The last bit of notation we need to get started is how we denote vectors. A vector $\boldsymbol{v} \in D \subseteq \mathbb{C}^n$ is denoted with boldface if we know it is in $D \subseteq \mathbb{C}^n$, and with normal font if it can be an infinite dimensional vector. There is also one set of vectors, $\mathbb{S}^{n-1}$, that will be used a lot in the section about CT image reconstruction. This set of vectors $\boldsymbol{v} \in \mathbb{R}^n$ such that $||\boldsymbol{v}||_2 = 1$.

2 From linear algebra to functional analysis

Functional analysis is an extremely useful tool in applied mathematics, where linear algebra framework is used to understand the structure of functions and operators. This has a wide range of applications ranging from integral equations to image compression and, as used in this text, image reconstruction.

2.1 Function spaces

We need a couple of definitions and clarifications in order to do our analysis, the first is what a **function space** is. A function space is simply a vector space where some of the vectors are functions. We also need to remember the definition of a basis.

Definition 2.1 (Basis)**.** A **basis** of a function space $\mathcal{X}$ is a set of functions $\mathscr{B} = \{x_i\}$ with $x_i \in \mathcal{X}$ and $i \in \mathbb{Z}_n$ for some $n \leq \infty$, such that for any $y \in \mathcal{X}$ there exists a set of $a_i \in \mathbb{C}$ such that $y = \sum_i a_i x_i$. We say that the function space $\mathcal{X}$ is **spanned** by the functions in $\mathscr{B}$, $\mathcal{X} = span(\mathscr{B})$.

Using this definition we define a basis for the space of quadratic functions

Example 2.1.1. The set of all quadratic functions can be described as functions in the function space $\mathcal{P}_2$, where $\mathcal{P}_2 = span(1, x, x^2)$.

This example shows how we can regard a quadratic function as a point in a three dimensional space. One axis is the constant coefficient, one is the linear coefficient and one is the quadratic coefficient. Another important tool when discussing function spaces is a space's dimension, which is defined as.

Definition 2.2 (Dimension of a function space)**.** The **dimension** of a function space $\mathcal{X}$, with basis $\mathscr{B}$, denoted $dim(\mathcal{X})$ is given by the order of $\mathscr{B}$.

$$dim(\mathcal{X}) = |\mathscr{B}| \,. \tag{2.1}$$

One way of interpreting the dimension of a function space $\mathcal{X}$ is how many free parameters can choose freely in order to uniquely define any element in $\mathcal{X}$. Going back to Example 2.1.1 shows that the dimension of the $\mathcal{P}_2$ space is 3, which makes sense since we know that any quadratic polynomial is uniquely given by three parameters.

The way a function space's dimension is defined reveals, the interesting fact, that function spaces can be infinite dimensional. That is, of course, if we are able to find infinitely many linearly independent functions, which we easily do by choosing polynomials.

Example 2.1.2. The set, $\mathcal{X}$, of all functions given by a linear combination of x raised to an arbitrary power n is a function space.

$$f(x) = \sum_{i=0}^{\infty} a_i x^i \tag{2.2}$$

This gives the rise to the basis $\mathscr{B} = \{1, x, x^2, ...\}$. We know from calculus that this is a linearly independent set that is closed under addition and scalar multiplication. We can therefore conclude that this is a function space spanned by $\mathscr{B}$. The cardinalty of $\mathscr{B}$, $|\mathscr{B}| = \aleph_0$ so the dimension is ∞, which proves that there exists function spaces with infinite dimension.

2.2 Norms and inner products

Norms are a very powerful tool that is used as a generalisation of distance for vector and function spaces.

Definition 2.3 (Norms)**.** A **normed function space** is a function space $\mathcal{X}$ equipped with a mapping $||\bullet||$ called a **norm**. Any norm satisfies these conditions

1. A norm has real range
$$||\bullet|| : \mathcal{X} \to \mathbb{R}. \tag{2.3}$$
2. The norm of any vector is non-negative
$$||v|| \geq 0. \tag{2.4}$$
3. Separation of points
$$||v|| = 0 \text{ iff } v = 0. \tag{2.5}$$
4. Scaling of vectors corresponds to scaling of norm
$$||av|| = |a| \ ||v||. \tag{2.6}$$
5. The triangle inequality
$$||v + u|| \leq ||v|| + ||u||. \tag{2.7}$$

With $v, u \in \mathcal{X}$ and $a \in \mathbb{R}$.

There is a notable extension of the norm, namely seminorms.

Definition 2.4 (Seminorms)**.** A **seminorm** is a mapping that satisfies all conditions but (2.5) of a norm.

It is important to note that a seminorm can no longer be regarded as a length measure, since the condition 2.5 is the condition that separates vectors. That means that two vectors that have different "coordinates" have no distance between them. Something that doesn't make sense according our notion of distance. The next step towards an understanding of functional analysis is some generalisation of angle and this comes on the form of inner products.

Definition 2.5 (Inner Products)**.** An **inner product** in the function space $\mathcal{X}$ is a function $\langle \cdot, \cdot \rangle : \mathcal{X}^2 \to \mathbb{C}$. The inner product function needs to satisfy these conditions:

1. Positive Definiteness

$$\langle x, x \rangle \geq 0 \, \langle x, x \rangle = 0 \Leftrightarrow x = 0. \tag{2.8}$$

2. Conjugate symmetry

$$\langle x, y \rangle = \overline{\langle y, x \rangle}. \tag{2.9}$$

3. Linearity of first component

$$\langle ax + by, z \rangle = \langle ax, z \rangle + \langle by, z \rangle \, . \tag{2.10}$$

With $x, y, z \in \mathcal{X}$ and $a, b \in \mathbb{C}$ [17, p. 205] □

It is also important to note that any inner product space is also a normed space and the norm is defined as

$$||x||^2 = \langle x, x \rangle \, . \tag{2.11}$$

The previous two definitions are known for anyone who is familiar with more than just very basic linear algebra. There is however one key piece of information that usually is left out of these modules; there is nothing confining them to finite dimensional space. Once we have defined inner products we can define orthogonality.

Definition 2.6 (Orthogonality)**.** We say that two functions x, y in some function space $\mathcal{X}$ with inner product $\langle \cdot, \cdot \rangle$ are **orthogonal**, $x \perp y$ if and only if

$$\langle x, y \rangle = 0. \tag{2.12}$$

The **orthogonal complement** $\mathcal{Y}^\perp$ of a subspace $\mathcal{Y} \subseteq \mathcal{X}$ is the set of functions

$$\mathcal{Y}^\perp = \{y \in \mathcal{X} : \langle y, x \rangle = 0 \forall x \in \mathcal{Y}\}. \tag{2.13}$$

There are two very important categories of function spaces, Banach and Hilbert spaces, but before we can define those we need the definition of completeness.

Definition 2.7 (Cauchy sequence)**.** Given a function space $\mathcal{X}$ equipped with the norm $||\bullet||$, we say that the sequence $\{x_k \in \mathcal{X}\}$ is a **Cauchy sequence** if the difference of two subsequent elements decrease in such a way that $||x_n - x_{n-1}|| < \epsilon$ for any $\epsilon > 0$ and all n greater than some m.

The definition of closure and complete function spaces comes straight out from this definition

Definition 2.8 (Closure)**.** Given a function space $\mathcal{X} \subseteq \mathcal{Y}$ where $\mathcal{Y}$ is complete. We then say that the closure of $\mathcal{X}$, $\bar{\mathcal{X}}$ is the set of all limit of all Cauchy sequences in $\mathcal{X}$.

$$\overline{\mathcal{X}} = \{ \lim_{k\to\infty} \{x_k\} : \{x_k\} \text{ is a Cauchy sequence and } x_k \in \mathcal{X} \forall k \in \mathbb{N}\}. \tag{2.14}$$

We say that a function space $\mathcal{X} \subseteq \mathcal{Z}$ is closed in $\mathcal{Z}$ if its closure $\overline{\mathcal{X}} \subseteq \mathcal{Z}$.

Definition 2.9 (Complete function space)**.** A function space $\mathcal{X}$, with norm $||\bullet||$ is complete if any Cauchy sequence $\{x_k \in \mathcal{X}\}$ converges to a function $x^\star \in \mathcal{X}$. Another way of saying this is $\overline{\mathcal{X}} = \mathcal{X}$.

It can be shown that it is impossible to come up with non-converging Cauchy sequences in finite dimensional normed spaces, so coming with an example that both describes the space and that are linear is unfortunately very difficult. We will therefore have two examples of Cauchy sequences, one to describe the how we can imagine an incomplete space and one of a nonconverging sequence of functions.

Example 2.2.1. Let $\mathbb{R}^*$ be the set of all non-zero numbers and the sequence $\{x_k\}$ be defined as $x_k = e^{-k}$ for any $k \in \mathbb{N}$. This sequence converges to 0, but $0 \notin \mathbb{R}^*$ so $\mathbb{R}^*$ is necessarily closed.

This example doesn't hold for function spaces, but it helps to give an understanding of what completeness means in the topological spaces. Namely that one can imagine certain points "missing" from the space. The next example will show how we can construct a nonconverging function sequence

Example 2.2.2. Let $f_k \in C([-1,1])$, the set of all continuous functions with domain $\mathcal{D}(F_k) = [0,1]$. Further more let k be an integer and the sequence $\{f_k(x)\}$ be given by

$$f_k(x) = \begin{cases} 0 & , -1 \leq x \leq 0 \\ ax & , 0 < x < \frac{1}{a} \\ 1 & , \frac{1}{a} \leq x \leq 1 \end{cases}. \tag{2.15}$$

This sequence is a Cauchy sequence in $C([-1,1])$ with respect to the L_1 norm, $||f_k||_1 = \int_{-1}^{1} f_k(x)dx$, that doesn't converge.

Proof. To prove whether or not this is a Cauchy sequence we need to prove that $f_{k+1} - f_k \to 0$, and the way we do that is

$$f_{k+1} - f_k = \begin{cases} 0 & ,-1 \leq x \leq 0 \\ (a+1-a)x = x & ,0 < x < \frac{1}{a+1} \\ 1 - ax & ,\frac{1}{a+1} \leq x < \frac{1}{a} \\ 0 & ,\frac{1}{a} \leq x \leq 1 \end{cases}. \tag{2.16}$$

Solving this for the L_1 norm gives

$$||f_{k+1} - f_k||_1 = \int_0^{\frac{1}{a+1}} |x|dx + \int_{\frac{1}{a+1}}^{\frac{1}{a}} |1 - ax|dx \tag{2.17}$$

Solving these integrals gives

$$||f_{k+1} - f_k||_1 = \frac{1}{2a(a+1)}. \tag{2.18}$$

We know that $2a(a+1) \to \infty$ as $a \to \infty$ which means that $\frac{1}{2a(a+1)} \to 0$ when $a \to \infty$. Next up is proving that it converges to a function $f^\star(x) \notin C([-1,1])$, letting $a \to \infty$ gives

$$f^\star(x) = \begin{cases} 0 & ,-1 \leq x \leq 0 \\ lim_{a\to\infty} ax & ,0 < x < lim_{a\to\infty} \frac{1}{a} \\ 1 & ,lim_{a\to\infty} \frac{1}{a} \leq x \leq 1 \end{cases}. \tag{2.19}$$

Setting inserting $lim_{a\to\infty} = 0$ gives

$$f^\star(x) = \begin{cases} 0 & ,-1 \leq x \leq 0 \\ 1 & ,0 \leq x \leq 1 \end{cases} \tag{2.20}$$

which is discontinuous in $x = 0$, and thus not in $C([-1,1])$. This concludes the proof that $\{f_k\}$ is a nonconverging Cauchy sequence in $C([-1,1])$ with respect to the L_1 norm. □

We can finally define Banach and Hilbert spaces, now that these concepts are defined.

Definition 2.10 (Banach and Hilbert spaces)**.** A function space is a **Banach space** equipped with a norm, and is complete with respect to that norm. It is a **Hilbert space** if the norm comes from an inner product, such as in (2.11).

We also have this important theorem

Theorem 2.1
Let $\mathcal{X}$ be a Hilbert space, and suppose $\mathcal{Y} \subseteq \mathcal{X}$ is also a Hilbert space. We then have that

$$\overline{\mathcal{Y}} \oplus \mathcal{Y}^\perp = \mathcal{X}. \tag{2.21}$$

The proof of this theorem is a bit outside the scope of this project, but is important nonetheless.

Example 2.2.3. The function $\langle x, y\rangle = \int_{-\infty}^{\infty} x(t)y(\bar{t})dt$ is an inner product in the function space L_2
Proof:

1. First we test positive definiteness

$$\langle x, x\rangle = \int_{-\infty}^{\infty} x^2(t)dt. \tag{2.22}$$

   We know from calculus that this is non-negative for any function x with $\mathcal{R}(x) \subseteq (R)$ and only equal to 0 if $x(t) = 0$. That proves the first condition of an inner product

2. Secondly the symmetry needs to be proven

$$\int_{-\infty}^{\infty} x(t)y(t)dt = \int_{-\infty}^{\infty} y(t)x(t)dt = \int_{-\infty}^{\infty} \overline{y(t)}x(t)dt. \tag{2.23}$$

   This proves the second condition, as long as $\mathcal{R}(x) \subseteq (R)$ for all $x \in \mathcal{X}$.

3. And lastly the linearity needs to be proven.

$$\int_{-\infty}^{\infty} (ax(t) + by(t))z(t)dt = \int_{-\infty}^{\infty} ax(t)z(t)dt + \int_{-\infty}^{\infty} by(t)z(t)dt. \tag{2.24}$$

   This is a well known fact from calculus and proves that this is an inner product

   It is also possible to show that the function $||x|| = \sqrt{\langle x, x\rangle}$ is a norm, this norm is called the L_2 norm.

This example shows that inner products and norms in infinite dimensional vector spaces are more than just hypothetical, we can actually create these functions.

2.2.1 Important norms

This section will introduce some important norms that will be used a lot later in this text.

2.2.2 The p-norms

The p-norms are norms for vectors in $\mathbb{R}^n$ and are as follows:

$$||\boldsymbol{x}||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}. \tag{2.25}$$

The most important p-norms are the 2-norm (the Euclidean norm) and the 1-norm (the Manhattan norm).

$$||\boldsymbol{x}||_1 = \sum_{i=1}^{n} |x_i| \tag{2.26}$$

$$||\boldsymbol{x}||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}. \tag{2.27}$$

L_p Norms

The L_p norms are a generalisation of the p-norms. The L_p is to create Banach Spaces with domain D, and are defined as follows:

$$||x||_p = \sqrt[p]{\int_D |x(t)|^p dt} \tag{2.28}$$

2.2.3 The ∞-norm

The ∞-norm or supremum norm of a vector in $\mathbb{R}^n$ or a function is the maximum absolute value of an entry in the vector, or the supermum of the absolute value of the function.

2.2.4 The total variation seminorm

The total variation (TV) seminorm is important in image denoising [2] and is, in the continuous case defined as:

$$||x||_{TV} = ||\nabla x||_1. \tag{2.29}$$

This norm is, as opposed to the L_p norms, not easy to discretise. We want a discretisation that works for images, or two dimensional functions that are piecewise constant in a square grid. There are multiple ways of discretising the TV-norm, and we will use the *isotropic* discretisation in this text, given by: [2]

$$\begin{aligned} ||\boldsymbol{x}||_{TV_I} =& \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} \sqrt{(\boldsymbol{x}_{i+1,j} - \boldsymbol{x}_{i,j})^2 + (\boldsymbol{x}_{i,j+1} - \boldsymbol{x}_{i,j+1})^2} \\ &+ \sum_{i=1}^{n-1} |\boldsymbol{x}_{i+1,m} - \boldsymbol{x}_{i,m}| + \sum_{j=1}^{m-1} |\boldsymbol{x}_{n,j+1} - \boldsymbol{x}_{n,j}|. \end{aligned} \tag{2.30}$$

2.3 Different function spaces

There are many different function spaces, and this section is here to define a couple of important function spaces.

2.3.1 The n-differentiable continuous function spaces

The function space of all continuous functions with domain D are denoted $C(D)$. All n-times continuously differentiable functions also form a function space and are denoted $C^n(D)$. These function spaces can be extended to work for all functions with compact support. That is denoted $C_0(D)$ for continuous functions with compact support and $C_0^n(D)$ for n-times differentiable functions with compact support.

2.3.2 Norm-induced function spaces

One of the reasons to talk about norms is that we can define the function space generated by a norm. We do this in order to get the function space containing all function having some wanted property.

Theorem 2.2
Let $\mathcal{X}$ be some function space, and let $||\bullet|| : \mathcal{X} \to \mathbb{R}$. We can say that the norm $||\bullet||$ generates a function space $\mathcal{Y}$ with the following definition:

$$\mathcal{Y} = \{x \in \mathcal{X} : ||x|| < \infty\}. \tag{2.31}$$

Proof. $\mathcal{X}$ is a function space by assumption, so all the properties of a function space but closedness are already fulfilled. The following two conditions are required to prove closedness.

$$||x + y|| < \infty \tag{2.32}$$
$$||ax|| < \infty. \tag{2.33}$$

Where $x, y \in \mathcal{Y}$ and $a \in \mathbb{R}$. The proof of condition (2.32) comes from the triangle inequality:

$$||x + y|| \leq ||x|| + ||y|| < \infty. \tag{2.34}$$

Where the last bit of this inequality comes from the definition of x and y and the fact that the sum of two real numbers can't be infinite. The second condition comes straight out of scaling of vectors.

$$||ax|| = |a|\ ||x|| < \infty. \tag{2.35}$$

The last bit here also comes from the definition of our vector space, and the fact that one can never get infinity by multiplying with a real number. □

Schwartz spaces

Schwartz spaces are a very important set of function spaces and will be integral in our understanding of image reconstruction.

Definition 2.11 (Schwartz space)**.** The **Schwarts space** over D the function-space of functions $f : D \to \mathbb{R}$ with the property that

$$(x_1^{\alpha_1}...x_n^{\alpha_n})(\partial_{x_1}^{\beta_1}...\partial_{x_n}^{\beta_n})f(x_1,...,x_n) < C \tag{2.36}$$

where $\partial_{x_i}^l f$ is the lth partial derivative of f with respect to x_i, α_i and β_i are arbitrary integers and $C \in \mathbb{R}$.

This can informally be said as the functions f where all derivatives of f tends to zero faster than any possible polynomial grows.

2.4 Linear operators

Linear operators are generalisations of matrices from finite dimensional spaces so as to work in infinite dimensional function spaces. A linear operator is defined as.

Definition 2.12 (Linear operators)**.** Given the function spaces $\mathcal{X}$ and $\mathcal{Y}$. A **linear operator** is a mapping with this property:

$$L(ax + by) = aLx + bLy \tag{2.37}$$

for any $x, y \in \mathcal{X}$ and any $a, b \in \mathbb{R}$.

The analysis of linear operators are very important, as they are found in a massive spectrum of problems. We will later see that CT-image reconstruction involves linear operators, but for now this example from quantum physics will be used.

Example 2.4.1. The energy operator $\hat{H}$ from quantum physics is a linear operator

$$\hat{H}\Psi(x) = \frac{-\hbar}{2}\nabla^2\Psi(x) + V(x)\Psi(x) \tag{2.38}$$

where Ψ is a function $\Psi : \mathbb{R}^3 \to \mathbb{C}$ and $||\Psi|| = 1$, and $V(x)$ is a function $V : \mathbb{R}^3 \to \mathbb{R}$. Showing that this is a linear opearator is easily done by testing the property (2.37).

$$\hat{H}(a\Psi(x) + b\Phi(x)) = \frac{-\hbar}{2}\nabla^2(a\Psi(x) + b\Phi(x)) + V(x)(a\Psi(x) + b\Phi(x)) \tag{2.39}$$

$$= a(\frac{-\hbar}{2}\nabla^2\Psi(x) + V(x)\Psi(x)) + b(\frac{-\hbar}{2}\nabla^2\Phi(x) + V(x)b\Phi(x)) \tag{2.40}$$

$$= a\hat{H}\Psi(x) + b\hat{H}\Phi(x). \tag{2.41}$$

This concludes the proof that the energy operator is a linear operator.

There are a couple of other important concepts that need to be generalised from linear algebra. The first is the transpose of a real matrix. The transpose has the property that $Ax \cdot y = x \cdot A^T y$. We define the adjoint of a linear operator the same way.

Definition 2.13 (Adjoint operators)**.** The **adjoint** L^* of a linear operator $L : \mathcal{X} \to \mathcal{Y}$ between Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$ is defined as the linear operator with these properties

$$L^* : \mathcal{Y} \to \mathcal{X} \tag{2.42}$$
$$\langle Lx, y\rangle = \langle x, L^* y\rangle \tag{2.43}$$

for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

The next theorem is extremely useful and is easily recognisable from linear algebra. The proof of this theorem is outside the scope of this text, but it is important enough to be given space in this text.

Theorem 2.3
Given two Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, and the linear operator $L : \mathcal{X} \to \mathcal{Y}$

$$\overline{\mathcal{R}(L)} = ker(L^*)^\perp \tag{2.44}$$
$$\overline{\mathcal{R}(L^*)} = ker(L)^\perp \tag{2.45}$$

2.4.1 Difficulties in infinite dimensions

The next difficulty that arises in functional analysis is, as the attentive reader might have guessed from Theorem 2.3, that the range $\mathcal{R}(A)$ of a linear operator $A : \mathcal{X} \to \mathcal{Y}$ between Banach spaces is not necessarily a closed subspace in $\mathcal{Y}$. Another difficulty that arises is that linear operators between infinite dimensional spaces don't need to be continuous. This rises the question of what continuity is in infinite dimensions.

Definition 2.14 (Continuous linear operators)**.** A linear operator $A : \mathcal{X} \to \mathcal{Y}$ between two Banach spaces are said to be continuous if there exists an $\epsilon > 0$ for any $\delta > 0$ such that

$$||Ax|| < \delta \tag{2.46}$$

for all x s.t. $||x|| < \epsilon$.

This definition of continuity is the same as for one dimensional functions, except for it being norms instead of absolute value. It can furthermore be shown that continuity of operators between infinite dimensional spaces can depend on what norm one uses.

2.5 Singular value decomposition

Singular value decomposition (SVD) is a very important tool in linear algebra, so one can wonder whether or not something equivalent exists in functional analysis. And the answer to that is yes, but not for all linear operators, and only in Hilbert spaces. Before the singular value decomposition is discussed we need to define eigenfunctions and eigenvalues.

Definition 2.15 (Eigenvalues and eigenfunctions)**.** Given a self adjoint linear operator A, the **eigenvalues** and **eigenfunctions** are respectively the scalars λ_i and nontrivial functions f_i satisfying these conditions

$$Af_i = \lambda_i f_i. \tag{2.47}$$

We still need one definition in order to define the SVD of an operator, namely compactness.

Definition 2.16 (Compact operator [6])**.** A linear operator, $A : \mathcal{X} \to \mathcal{Y}$, between Banach spaces is said to be compact if the subsequence $\{A_{x_k}\}$ of a bounded sequence $\{x_k\}$ with $||x_k|| < M \in \mathbb{R}$ is a Cauchy sequence.

Definition 2.17 (Singular value decomposition)**.** The **singular value decomposition** of a compact operator $K : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are Hilbert spaces is given by

$$Kf = \sum_{i=1}^{n} u_i \sigma_i \langle v_i, f \rangle . \tag{2.48}$$

Here v_i are the eigenfunctions of the self adjoint operator K^*K, σ_i are the nonzero eigenvalues corresponding to v_i, and $u_i = Kv_i$. All functions are normalised and have norm equal to 1. The singular values σ_i are furthermore ordered such that $\sigma_i \geq \sigma_{i+1}$.

Proofs about the existence of a singular value decomposition are outside the scope of this project, but it is a very powerful tool that is important for the study of inverse problems.

3 Inverse Problems

3.1 What is an inverse problem

The study of inverse problems is concerned with calculating some initial conditions or parameters of a model from observed data. These problems span a wide range of problems, from CT image reconstruction to finding the heat resistance in a medium. [5] It therefore makes sense to get some rigorous mathematical background of this field, but some definitions must be in place before that is possible.

Definition 3.1 (Inverse problems)**.** Given two Banach spaces, $\mathcal{X}$ and $\mathcal{Y}$, an operator $A : \mathcal{X} \to \mathcal{Y}$ and some measured response data y. The problem:
Find x s.t.

$$Ax = y \tag{3.1}$$

is called an **inverse problem**.

We separate inverse problems into two different categories, well-posed problems and the ill-posed problems.

Definition 3.2 (Well- and ill-posed problems)**.** An inverse problem is **well-posed** if it satisfies all of these criteria

1. There is a solution for all admissible data
2. All solutions are unique
3. The solution depends continuously on the data

All problems that aren't well-posed are called **ill-posed**.

[5, p. 32]

We are in this text only interested in the ill-posed problems, as CT reconstruction is ill-posed. Ill posed problems come in many shapes and forms, and here are examples of two different ill-posed inverse problems.

Example 3.1.1. (Ill-posed inverse problems)
1. Find $\boldsymbol{x}$ such that:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.2}$$

This problem, as many might recognise comes up in linear regression, and is an ill-posed problem. We are, in this case, usually is interested in finding the least square solution to the system.

2. Find $\boldsymbol{x}$ such that:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.3}$$

This problem is ill-posed as it doesn't satisfy the second condition of Definition 3.2. We will later see that the discrete CT-problem is a problem on this form and one difficulty is finding a way to choose the correct solution.

One way to solve ill-posed inverse problems is finding the *best approximate solution* of (3.1), which is defined as.

Definition 3.3 (Best approximate solution)**.** The **best approximate solution** $x^\dagger$ of the problem 3.1 is the value

$$x^\dagger = \arg\min_{x \in \mathcal{F}} ||x||_2. \tag{3.4}$$

Where $\mathcal{F}$ is defined as

$$\mathcal{F} = \{\arg\min_{x \in \mathcal{X}} ||Ax - y||\}. \tag{3.5}$$

The best approximate solution to a system is therefore the least squares solution if there is no solution to the problem, and the least norm solution if there are multiple solutions to the system. To find this best approximate solution we introduce the *generalised inverse*.

Definition 3.4 (Generalised inverse)**.** The **generalised inverse**, or Moore-Penrose generalised inverse, of an operator $A : \mathcal{X} \to \mathcal{Y}$ is the operator $A^\dagger$ with the property:

$$A^\dagger y = x^\dagger. \tag{3.6}$$

The operator $A^\dagger$ can be expressed in terms of the singular value decomposition for compact linear operators.

Theorem 3.1 (Generalised inverse expressed in terms of SVD)
Given a compact linear operator K. The generalised inverse $K^\dagger$ is given by.

$$K^\dagger x = \sum_{i=1}^{\infty} v_i \sigma_i^{-1} \langle u_i, x \rangle . \tag{3.7}$$

Where u_i, σ_i and v_i are given by the singular value decomposition.

$$Kx = \sum_{i=1}^{\infty} u_i \sigma_i \langle v_i, x \rangle . \tag{3.8}$$

From this expression we see that the noise sensitivity, and thus how ill-posed it is, can be seen from the singular values of the operator we want to invert. The problem is severely ill-posed if they tend to zero "very quickly", whereas if they tend to zero "slowly" the problem is less affected by measurement errors, and thus only mildly ill-posed.

3.2 Regularisation techniques

There are unfortunately a couple of difficulties that arise in infinite dimensional space that doesn't arise in finite dimensions, both regarding noise. These problems are explained very well in the book Regularization of Inverse Problems by Engl, et al. [5]. The way it is done is by introducing $y^\delta \in \{y^\delta \in \mathcal{Y} : ||y - y^\delta|| \leq \delta\}$. The first difficulty that arises is that $A^\dagger$ doesn't need to be bounded, which means that $||A^\dagger||$ doesn't necessarily converge to a real number. There might therefore be two values y and y^δ with the property that $||A^\dagger y - A^\dagger y^\delta|| \gg 0$ even for very small δ. The next difficulty is that y^δ might not be in the range of A, $\mathcal{R}(A)$, but still be in the closure of the range of A, $\overline{\mathcal{R}(A)}$. This means that $A^\dagger$ isn't necessarily defined for y^δ. The way we cope with these problems is by introducing something called the regularisation of A.

Definition 3.5 (Regularisation techniques)**.** A **regularisation** T_λ of A is a family of continuous linear operators with closed range and the property that $T_\lambda \to A^\dagger$ as $\lambda \to 0$

There are many different regularisation techniques that are suitable for different types of problems. The first one we will look at is the most basic one, the truncated SVD.

3.2.1 Truncated singular value decomposition

The truncated SVD is a regularisation technique that works for any compact linear operator and is defined as.

Definition 3.6 (Truncated SVD)**.** Let $A : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are Hilbert spaces. The truncated SVD regularised inverse is then given by:

$$T_\lambda b = \sum_{i=1}^{\sigma_i > \lambda} v_i (\sigma_i)^{-1} \langle u_i, b \rangle . \tag{3.9}$$

This regularisation technique is not often used, and there are three important factors that makes this technique worse than other regularisation techniques. Firstly that the singular value decomposition of an operator might be very difficult to compute, socondly, calculating the sum of integrals is computationally heavy and lastly, that a lot of information is lost this way. Just omitting the most influential singular values isn't necessarily a good way of enforcing boundedness on the operator. To deal with those problems we introduce the Tikhonov regularisation

3.2.2 Tikhonov regularisation

Tikhonov regularisation is one of the most well known regularisation techniques, and is given by a penalty term for the L_2 norm.

Definition 3.7 (Tikhonov Regularisation)**.** Let $A : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are Hilbert spaces. The Tikhonov-regularized inverse of A is then given by:

$$T_\lambda b = \arg\min_x \{||Ax - b|| + \lambda||x||\}. \tag{3.10}$$

Studying the Tikhonov regularised inverse gives some understanding as to the inner structure of it. That structure can be seen by comparing the SVD of T_λ with the SVD of $A^\dagger$. We need this lemma from Natterer [10] before we can derive the singular values of T_λ.

Lemma 3.2 ([10])
The operator T_λ can be written in the following manner:

$$T_\lambda = (A^*A + \lambda I)^{-1}A^*. \tag{3.11}$$

Theorem 3.3
Let A be given by the SVD $Ax = \sum_i u_i\sigma_i \langle v_i, x\rangle$. The operator T_λ will then have the singular values ξ_i^λ:

$$\xi_i^\lambda = \frac{\sigma_i}{\sigma_i^2 + \lambda}. \tag{3.12}$$

Proof. This proof comes from finding the singular value decomposition of equation (3.11). We know from [5] that

$$A^*Ax = \sum_{i=1}^{\infty} v_i\sigma_i^2 \langle v_i, x\rangle . \tag{3.13}$$

We also know that the identity operator can be written as

$$Ix = \sum_{i=1}^{\infty} f_i \langle f_i, x\rangle . \tag{3.14}$$

For any complete orthonormal set of functions $\{f_i\}$ that can be used as a basis for $\mathcal{X}$. Hence the singular functions $\{v_i\}$ is such a set[5].

$$(A^*A + \lambda I)x = \sum_{i=1}^{\infty} v_i\sigma_i^2 \langle v_i, x\rangle + \lambda\sum_{i=1}^{\infty} v_i \langle v_i, x\rangle . \tag{3.15}$$

Combining the sums gives

$$(A^*A + \lambda I)x = \sum_{i=1}^{\infty} v_i(\sigma_i^2 + \lambda) \langle v_i, x\rangle . \tag{3.16}$$

Inverting this operator gives

$$(A^*A + \lambda I)^{-1}x = \sum_{i=1}^{\infty} v_i(\sigma_i^2 + \lambda)^{-1} \langle v_i, x\rangle . \tag{3.17}$$

We then substitute $(A^*A + \lambda I)^{-1}$ with this in (3.11) and get

$$(A^*A + \lambda I)^{-1} A^* x = \sum_{i=1}^{\infty} v_i (\sigma_i^2 + \lambda)^{-1} \langle v_i, A^* x \rangle . \tag{3.18}$$

The next step here is figuring out another expression for $\langle v_i, A^* x \rangle$. That starts with the singular value decomposition of $A^* x$

$$A^* x = \sum_{j=1}^{\infty} v_j \sigma_j \langle u_j, x \rangle . \tag{3.19}$$

Using this fact gives

$$\langle v_i, A^* x \rangle = \left\langle v_i, \sum_{j=1}^{\infty} v_j \sigma_j \langle u_j, x \rangle \right\rangle . \tag{3.20}$$

$\sigma_i \langle u_i, x \rangle$ are both scalars and can be moved out of the inner product, we also have linearity of the inner product, so the sum can also be moved out.

$$\langle v_i, A^* x \rangle = \sum_{j=1}^{\infty} \langle v_i, v_j \rangle \sigma_j \langle u_j, x \rangle . \tag{3.21}$$

Orthonormality of the singular functions gives $\langle v_i, v_j \rangle = 1$ for $i = j$ and 0 otherwise so the sum diminishes completely.

$$\langle v_i, A^* x \rangle = \sigma_i \langle u_i, x \rangle . \tag{3.22}$$

Using this equality in (3.18) gives the singular value decomposition of T_λ, and thus its singular values.

$$(A^*A + \lambda I)^{-1} A^* x = \sum_{i=1}^{\infty} v_i (\sigma_i^2 + \lambda) \sigma_i \langle u_i, x \rangle \tag{3.23}$$

$$= \sum_{i=1}^{\infty} v_i \frac{\sigma_i}{\sigma_i^2 + \lambda} \langle u_i, x \rangle . \tag{3.24}$$

□

Figure 3.1 gives a good understanding of what Tikhonov regularisation does. It shows that Tikhonov regularisation limits the amount that the magnitude of a signal can increase when we invert it. If we compare it to truncated SVD we see that instead of ignoring the singular values that cause the unboundedness of the inverse we decrease them. We can understand it as devaluating the importance of those singular functions instead of just ignoring them completely as truncated SVD does, which might very well be a much better idea.

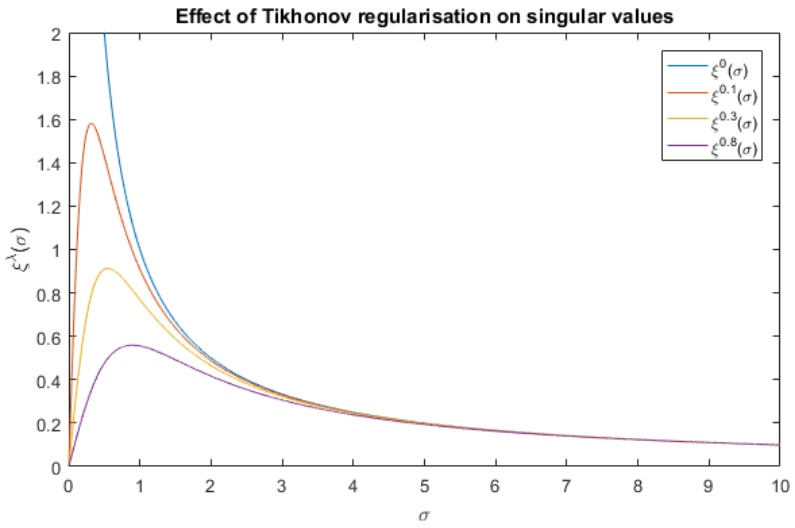

Figure 3.1: Effect of regularisation parameter for Tikhonov regularisation on it's singular values
The x-axis is the singular values of A, σ and the y-axis is the singular values of the regularised inverse T_λ, ξ^λ

3.2.3 L_1 regularisation and sparsity

We used an inner product induced norm in Tikhonov-regularisation, so some might ask the question, what happens if we use a norm without an inner product like the L_1 norm instead? Proving that this makes for a regularisation technique is outside the scope of this text, but L_1 regularisation is still important for inducing something called sparsity in the signal. We start by defining the L_1 regularisation technique.

Definition 3.8 (L_1 regularisation)**.** Let $A : \mathcal{X} \to \mathcal{Y}$ be a linear operator between the Banach spaces $\mathcal{X}$ and $\mathcal{Y}$ and define the operator $T_\lambda : \mathcal{Y} \to \mathcal{X}$ such that

$$T_\lambda b = \arg\min_x ||Ax - b||_2^2 + \lambda ||x||_1. \tag{3.25}$$

This operator is a regularisation technique for the operator A and is called the L_1 regularisation of A.

In real life we rarely calculate these problems with respect to operators and functions, but rather with matrices and vectors, a discrete approximation of the *actual* data and transformation. This technique will in that case induce something called *sparsity* in x. Namely that there are few nonzero elements in x [16]. This in itself isn't necessarily of any importance, but this gives rise to a very strong regularisation method if one combines it with some prior knowledge of the solution.

3.2.4 Sparsity regularisation

Sparsity regularisation is a concept from signal analysis where one uses some prior knowledge about the form of the signal in order to find a way to best describe it. An example of this can be from spectroscopy, where one knows that the signal is built up of a superposition of few sinusoidal functions. By using that knowledge, we can undersample our data and try to find the set of fewest different frequencies that describes the data. The most intuitive way of tackling this problem is trying to minimize the $L_0$0-seminorm of a vector, the number of nonzero entries. This problem is unfortunately NP-hard so to be able to solve the problem, we use the L_1 norm, which is the convex function that best approximates the L_0-norm.[16] Sparsity regularisation can therefore be done this way:

Definition 3.9 (Sparsity regularisation)**.** Let $A : \mathcal{X} \to \mathcal{Y}$ and define the operator $T_\lambda : \mathcal{Y} \to \mathcal{X}$ such that

$$T_\lambda b = \arg\min_x ||Ax - b||_2^2 + \lambda ||Dx||_1 \tag{3.26}$$

where D is an operator that changes the basis x is expressed by into a basis we assume is sparse. This operator is a regularisation technique for the operator A.

This technique might be difficult compute in practice since the $||D \bullet ||_1$ operator isn't differentiable, so we use the inverse map of D to solve it instead.

$$T_\lambda b = D^{-1} \arg\min_y ||AD^{-1}y - b||_2^2 + \lambda ||y||_1. \tag{3.27}$$

This is easier to solve, because we have good methods for minimizing the 1-norm of a vector. The difficulty in solving this problem is therefore to find the right basis that describes your signal, and choosing the right basis to describe tomographic X-ray images is still an open field of research.

3.2.5 Total variation regularisation

Total variation (TV) regularisation again builds on the same framework as Tikhonov and L_1 regularisation, but uses the TV-seminiorm as penalty term.

Definition 3.10 (TV-regularisation)**.** Let $A : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y}$ and $\mathcal{X}$ are Banach spaces and $\mathcal{Y} \subseteq C^1(\mathbb{R}^n)$ and define the operator $T_\lambda : \mathcal{Y} \to \mathcal{X}$ such that

$$T_\lambda b = \arg\min_x ||Ax - b||_2^2 + \lambda ||x||_{TV}. \tag{3.28}$$

This operator is a regularisation technique for the operator A, and is called the TV-regularisation of A.

We can regard TV-regularisation as a way to induce sparsity in the gradient of the function. This means that TV-regularisation tries to get piecewise constant regions of the function (where the gradient is 0) with sharp edges in order to minimise the residuals.

4 X-Rays and computerised tomography

4.1 Physical Background

X-rays are ionising electromagnetic waves in the energy spectrum $100eV - 100keV$ that often are used for medical imaging. That is because X-rays pass through soft tissue almost unaffected, whereas they are strongly weakened going through bone and cartilage.[14] The X-ray intensity loss is proportional with both length of tissue they pass through and the intensity of the X-ray beam resulting in the following differential equation:

$$dI = \mu I d\boldsymbol{l}. \tag{4.1}$$

Here I is the beam intensity, $d\boldsymbol{l}$ is the infinitesimal length of tissue the beam passes through, and μ is the attenuation coefficient of the tissue it passes through. See figure 4.1 for an illustration what the different parameters of our X-ray model are.

If we solve our differential equation with μ varying by $\boldsymbol{x} \in \mathbb{R}^2$ we get

$$\frac{I_0}{I} = e^{-\int_L \mu(\boldsymbol{x}) d\boldsymbol{x}} \tag{4.2}$$

where L is the line the beam passes through, I is the measured intensity after the beam has passed through the body, I_0 is the initial intensity measured beforehand without an object in the CT scanner and $\mu(\boldsymbol{x})$ is the attenuation coefficient of the tissue with coordinates $\boldsymbol{x} \in \mathbb{R}^2$. This model generates our inverse problem:

$$\begin{aligned} \text{Calculate: } & \mu(\boldsymbol{x}) \\ \text{Given: } & e^{-\int_{L_i} \mu(\boldsymbol{x}) d\boldsymbol{x}} \\ \text{For all lines in: } & \mathcal{L} = \{L_1, ..., L_n\} \end{aligned} \tag{4.3}$$

The set $\mathcal{L}$ contains many different lines with different angles and all lines with the same angle are called one tomographic slice. This is not a trivial problem to solve, so in order to do that we need some mathematical foundation.

4.2 Radon transformation

The Radon transform is the way we mathematically describes taking tomographic slices, and is defined as.

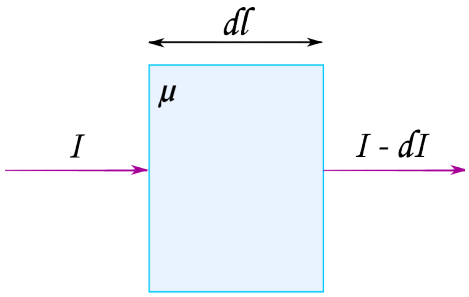

Figure 4.1: Figure explaining our model for X-ray intensity loss, dl is infinitesimal tissue length, μ is the attenuation coefficient of the tissue, I is X-ray intensity before passing through the tissue and dI is X-ray intensity loss after passing through the tissue

Definition 4.1 (Radon transform)**.** The operator $\mathscr{R} : \mathcal{S}(\mathbb{R}^2) \to \mathcal{S}(\mathbb{S}^1 \times \mathbb{R})$ is defined as

$$\mathscr{R}f(\boldsymbol{\theta}, r) = \int_{\boldsymbol{x}\cdot\boldsymbol{\theta}=r} f(x)dx = \int_{\theta^\perp} f(r\boldsymbol{\theta} + \boldsymbol{y})d\boldsymbol{y}. \tag{4.4}$$

Where $\boldsymbol{\theta} \in \mathbb{S}^1$ and $r \in \mathbb{R}$. The function $\mathscr{R}f$ is called a sinogram.

The Radon transform $\mathscr{R}f(\boldsymbol{\theta}, r)$ takes a function f and returns the value of its integral along the line perpendicular to $\boldsymbol{\theta}$ that goes through the point $r\boldsymbol{\theta}$. A geometrical image of what the parameters symbolise can be seen from Figure 4.2. It is important to note that the Radon transform returns a symmetric function, that is

Proposition 4.1 (Symmetry of $\mathscr{R}$)

$$\mathscr{R}f(\boldsymbol{\theta}, r) = \mathscr{R}f(-\boldsymbol{\theta}, -r). \tag{4.5}$$

Proof.

$$\mathscr{R}f(-\boldsymbol{\theta}, -r) = \int_{-\theta^\perp} f((-r)(-\boldsymbol{\theta}) + y)dy. \tag{4.6}$$

Using the fact that $\theta^\perp = -\theta^\perp$ we get

$$\mathscr{R}f(-\boldsymbol{\theta}, -r) = \int_{\theta^\perp} f(r\boldsymbol{\theta} + y)dy = \mathscr{R}f(\boldsymbol{\theta}, r). \tag{4.7}$$

□

This proposition makes sense from a purely physical point of view: the direction the beam is traveling shouldn't affect the observed data. It also implies that we don't need to rotate the beam a whole revolution in order to take a CT-scan but rather half a revolution. The next step in developing a way to reconstruct images is to introduce the *Radon slice operator*.

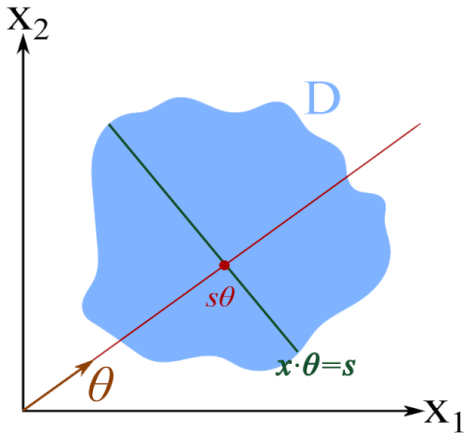

Figure 4.2: Illustration of the different parameters for the Radon transform on a function with compact support.

Definition 4.2 (Radon slice operator[10])**.** Let f be a function in $\mathcal{S}(\mathbb{R}^2)$ and $\boldsymbol{\theta} \in \mathbb{S}^1$ the **Radon slice operator** along the angle is then given by

$$\mathscr{R}_\theta f(r) = \mathscr{R} f(\boldsymbol{\theta}, r). \tag{4.8}$$

We say that $\mathscr{R}_\theta f(r)$ is the slice of f along $\boldsymbol{\theta}$.

This operator is how we model one projection, and will be important in understanding the adjoint operator of $\mathscr{R}$, but it might be a bit difficult to follow the upcomming proofs without some extra notation, we therefore introduce this notation:

$$\mathscr{R}_\theta f(r) = g(\boldsymbol{\theta}, r) = g^\theta(r). \tag{4.9}$$

We now want to calculate adjoints of these operators to use the framework we defined in chapter two and three, but before we can do that we need to prove that the Radon transform and the Radon slice operator are linear operators.

Proposition 4.2
The Radon transform $\mathscr{R} : \mathcal{S}(\mathbb{R}^2) \to \mathcal{S}(\mathbb{S}^1 \times \mathbb{R})$ and the radon slice operator $\mathscr{R}_\theta : \mathcal{S}(\mathbb{R}^2) \to \mathcal{S}(\mathbb{R})$ are linear operators

Proof. We will only prove that the Radon transform is linear since the proof is very similar for the slice operator. To prove this we need to prove that $\mathscr{R}(af + bg) = a\mathscr{R}f + b\mathscr{R}g$ with $f, g \in \mathcal{S}(\mathbb{R}^2)$, something that comes straight out

of its definition.

$$\begin{aligned}\mathscr{R}(af+bg) &= \int_{\boldsymbol{x}\cdot\boldsymbol{\theta}=r} (af(\boldsymbol{x})+bg(\boldsymbol{x}))d\boldsymbol{x} \\ &= a\int_{\boldsymbol{x}\cdot\boldsymbol{\theta}=r} f(\boldsymbol{x})d\boldsymbol{x} + b\int_{\boldsymbol{x}\cdot\boldsymbol{\theta}=r} g(\boldsymbol{x})d\boldsymbol{x} \\ &= a\mathscr{R}f(\boldsymbol{x}) + b\mathscr{R}g(\boldsymbol{x}). \end{aligned} \tag{4.10}$$

□

The next step after showing that the Radon transform is a linear transformation is finding its and the Radon slice operator's adjoints, but before that we need to define what inner product we are using. For $f_i \in \mathcal{S}(\mathbb{R}^2)$, $g_i \in \mathcal{S}(\mathbb{S}^1 \times \mathbb{R})$ and $g_i^\theta \in \mathcal{S}(\mathbb{R})$

$$\langle f_1, f_2\rangle = \int_{\mathbb{R}^2} f_1(\boldsymbol{x})f_2(\boldsymbol{x})d\boldsymbol{x} \tag{4.11}$$

$$\langle g_1, g_2\rangle = \int_{\boldsymbol{\theta}\in\mathbb{S}^1}\int_{r\in\mathbb{R}} g_1(\boldsymbol{\theta},r)g_2(\boldsymbol{\theta},r)drd\boldsymbol{\theta} \tag{4.12}$$

$$\left\langle g_1^\theta, g_2^\theta\right\rangle = \int_{r\in\mathbb{R}} g_1^\theta(r)g_2^\theta(r)dr. \tag{4.13}$$

Those inner products gives the adjoints

Proposition 4.3 ([10])
The adjoint of the Radon slice operator is

$$\mathscr{R}_\theta^* g(\boldsymbol{x}) = g(\boldsymbol{\theta}, \boldsymbol{\theta}\cdot\boldsymbol{x}). \tag{4.14}$$

And the adjoint of the Radon transform is

$$\mathscr{R}^* g(\boldsymbol{x}) = \int_{\boldsymbol{\theta}\in\mathbb{S}^1} g(\boldsymbol{\theta}, \boldsymbol{\theta}\cdot\boldsymbol{x})d\boldsymbol{\theta}. \tag{4.15}$$

Proof. The adjoint of these operators is as earlier defined as follows.

$$\langle \mathscr{R}_\theta f, g\rangle = \langle f, \mathscr{R}_\theta^* g\rangle \tag{4.16}$$

$$\langle \mathscr{R} f, g\rangle = \langle f, \mathscr{R}^* g\rangle . \tag{4.17}$$

We start by proving (4.16)

$$\begin{aligned}\langle \mathscr{R}_\theta f, g\rangle &= \int_{\mathbb{R}} \mathscr{R}_\theta f(r)g^\theta(r)dr \\ &= \int_{\mathbb{R}} \left[\int_{\theta^\perp} f(r\boldsymbol{\theta}+\boldsymbol{y})d\boldsymbol{y}\right] g^\theta(r)dr. \end{aligned} \tag{4.18}$$

Including $g^\theta(r)$ in the integral with respect to $\boldsymbol{y}$ yields

$$\langle \mathscr{R}_\theta f, g\rangle = \int_{\mathbb{R}}\int_{\theta^\perp} f(r\boldsymbol{\theta}+\boldsymbol{y})g^\theta(r)d\boldsymbol{y}dr. \tag{4.19}$$

We can combine these two integrals to be over the whole $\mathbb{R}^2$ plane since $\boldsymbol{x}$ is integrated over the line perpendicular to $\boldsymbol{\theta}$ that goes through $r\boldsymbol{\theta}$ and r is integrated over the whole real number line, so this we integrate over all possible lines in $\mathbb{R}^2$. We also substitute $g^\theta(r)$ with $g^\theta(\boldsymbol{x}\cdot\boldsymbol{\theta}$ and get

$$\langle \mathscr{R}_\theta f, g\rangle = \int_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) g^\theta(\boldsymbol{x}\cdot\boldsymbol{\theta}) d\boldsymbol{x} = \langle f, \mathscr{R}_\theta^* g\rangle \tag{4.20}$$

We now prove (4.17)

$$\begin{aligned}\langle \mathscr{R} f, g\rangle &= \int_{\boldsymbol{\theta}\in\mathbb{S}^1}\int_{r\in\mathbb{R}} \mathscr{R} f(\boldsymbol{\theta}, r) g(\boldsymbol{\theta}, r) dr d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\theta}\in\mathbb{S}^1}\int_{r\in\mathbb{R}} \left[\int_{\boldsymbol{x}\cdot\boldsymbol{\theta}=r} f(\boldsymbol{x}) d\boldsymbol{x}\right] g(\boldsymbol{\theta}, r) dr d\boldsymbol{\theta}\end{aligned} \tag{4.21}$$

Combining the inner two integrals as with $\mathscr{R}_\theta^*$ gives

$$\langle \mathscr{R} f, g\rangle = \int_{\boldsymbol{\theta}\in\mathbb{S}^1}\int_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) g(\boldsymbol{\theta}, \boldsymbol{x}\cdot\boldsymbol{\theta}) d\boldsymbol{x} d\boldsymbol{\theta}. \tag{4.22}$$

Changing the order of the integral concludes the proof

$$\langle \mathscr{R} f, g\rangle = \int_{\boldsymbol{x}\in\mathbb{R}^2}\int_{\boldsymbol{\theta}\in\mathbb{S}^1} f(\boldsymbol{x}) g(\boldsymbol{\theta}, \boldsymbol{x}\cdot\boldsymbol{\theta}) d\boldsymbol{\theta} d\boldsymbol{x} = \langle f, \mathscr{R}^* g\rangle \tag{4.23}$$

□

The adjoint Radon transform is called the backprojection operator and it is also interesting to note that $\mathscr{R}^* f = \int_{\mathbb{S}^1} \mathscr{R}_\theta^* f d\boldsymbol{\theta}$, the backprojection operator can in other words be seen as the sum of all possible adjoint slice operators. Understanding the adjoint slice operator will therefore give us an understanding of the backprojection operator. Figure 4.3 shows that the single-slice backprojection $\mathscr{R}_\theta^* g$ returns a two dimensional function, constant along $\boldsymbol{\theta}^\perp$ and equal to g along $\boldsymbol{\theta}$. Figure 4.4 might also help understanding, as it shows how the backprojection operator changes as one adds more and more projections.

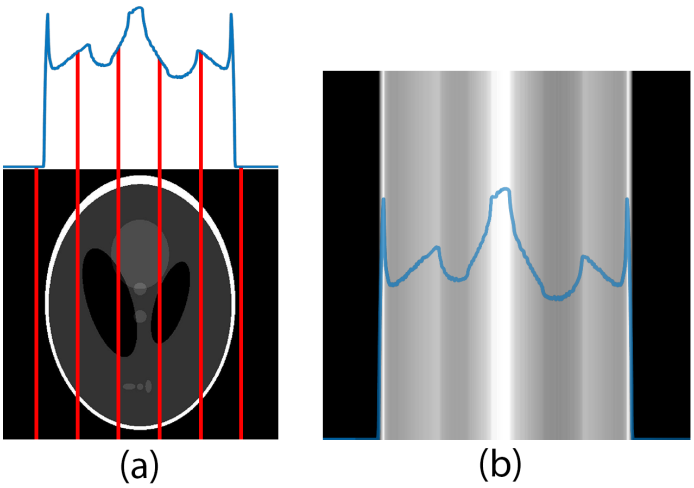

Figure 4.3: (a): Shows an image $f \in \mathcal{S}(R^2)$ where black is low values and white is high values. The blue graph is $\mathscr{R}_\theta f$ with $\boldsymbol{\theta} = (1,0)^T$ and the red lines shows some of the integral lines. (b): Shows $\mathscr{R}_\theta^* \mathscr{R}_\theta f$ with the same f and $\boldsymbol{\theta}$ as (a). The blue graph is $\mathscr{R}_\theta f$

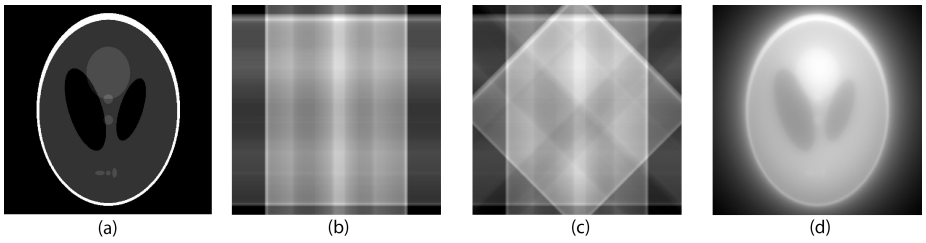

Figure 4.4: (a): Shows an image $f \in \mathcal{S}(\mathbb{R}^2)$ (b): Backprojection with two angles, $(\mathscr{R}_{\theta_1}^* \mathscr{R}_{\theta_1} + \mathscr{R}_{\theta_2}^* \mathscr{R}_{\theta_2}) f$. The angles are $\boldsymbol{\theta_1} = (1,0)^T$ and $\boldsymbol{\theta_3} = (0,1)^T$. (c): Backprojection with four angles. (d) Backprojection with all angles, $\mathscr{R}^* \mathscr{R} f$

.

We instantly see similarities between (a) and (d) in Figure 4.4, which leads the question of whether or not the backprojection operator can be used for reconstructing f from g where $\mathscr{R} f = g$. That is the case, and is called the filtered backprojection algorithm.

4.3 The Fourier Slice Theorem and the filtered backprojection reconstruction algorithm

Filtered backprojection might, very well, be the most well known inversion formula for CT-image reconstruction, and builds on the famous theorem Fourier Slice Theorem by Johan Radon.

Theorem 4.1 (Fourier Slice Theorem [10])
Let $f \in \mathcal{S}(\mathbb{R}^2)$, where $\mathcal{S}(\mathbb{R}^2)$ is the Schwartz space with functions over $\mathbb{R}^2$. We then have that

$$\mathscr{F}_r(\mathscr{R}_\theta f)(\sigma) = \sqrt{2\pi}\hat{f}(\sigma\theta) \qquad \text{with } \sigma \in \mathbb{R} \tag{4.24}$$

Proof.

$$\mathscr{F}_r(\mathscr{R}_\theta f)(\sigma) = (2\pi)^{-\frac{1}{2}} \int_{\mathbb{R}} e^{-i\sigma r} \mathscr{R}_\theta f(r) dr \tag{4.25}$$

$$= (2\pi)^{-\frac{1}{2}} \int_{\mathbb{R}} \int_{\boldsymbol{y}\in\boldsymbol{\theta}^\perp} e^{-i\sigma r} f(r\boldsymbol{\theta} + \boldsymbol{y}) d\boldsymbol{y} dr \tag{4.26}$$

We pick our axes freely as long as they are orthogonal, without changing the limits of the integral since integral is over the whole $\mathbb{R}^2$ plane. The axes we choose are such that $\boldsymbol{\theta} = (0,1)^T$, which gives $\boldsymbol{y} = (y_1, 0)$ by the fact that $\boldsymbol{y} \in \theta^\perp$. This changes the integrand to $dy_1 dr$ and we end up with an integral on this form

$$\mathscr{F}_r(\mathscr{R}_\theta f)(\sigma) = (2\pi)^{-\frac{1}{2}} \int_{r\in\mathbb{R}} \int_{y_1\in\mathbb{R}} e^{-i\sigma r} f(y_1 \boldsymbol{e_1} + r\boldsymbol{e_2}) dy_1 dr$$

$$= (2\pi)^{-\frac{1}{2}} \int_{\mathbb{R}^2} e^{-i\sigma r} f(\boldsymbol{x}) d\boldsymbol{x} \tag{4.27}$$

The vector $\boldsymbol{x}$ is here defined as $\boldsymbol{x} = (y_1, r)^T$. We now use the fact that $r = \boldsymbol{x} \cdot \boldsymbol{\theta}$ to get

$$\mathscr{F}_r(\mathscr{R}_\theta f)(\sigma) = (2\pi)^{-\frac{1}{2}} \int_{\mathbb{R}^2} e^{-i\sigma \boldsymbol{x}\cdot\boldsymbol{\theta}} f(\boldsymbol{x}) d\boldsymbol{x} \tag{4.28}$$

Using the definition of the two dimensional Fourier transform gives

$$\mathscr{F}_r(\mathscr{R}_\theta f)(\sigma) = \sqrt{2\pi}\hat{f}(\sigma\boldsymbol{\theta}) \tag{4.29}$$

With $\sigma \in \mathbb{R}$. □

A similar proof can be constructed in n-dimensional systems, where we have $\boldsymbol{y} = (y_1, ..., y_{n-1}, 0)^T$ and $\boldsymbol{\theta} = (0, ..., 0, 1)^T$. An interpretation of this theorem is that the Radon-slice of f along $\boldsymbol{\theta}$ gives the frequencies of f along θ in polar coordinates. This theorem gives might be the most important theorem in CT-reconstruction and is therefore often called the *Fundamental Slice Theorem*. It is this theorem that gives birth to this inversion formula:

Theorem 4.2 (Filtered backprojection)
Given $f \in \mathcal{S}(\mathbb{R}^2)$ and $\mathscr{R}f = g$.

$$f = \mathscr{R}^* \mathscr{F}^{-1} \frac{|\sigma|}{4\pi} \mathscr{F} g \tag{4.30}$$

Proof. We start this proof by representing f in respect to its Fourier transform

$$f(\boldsymbol{x}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} e^{i\boldsymbol{\xi}\cdot\boldsymbol{x}} \hat{f}(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{4.31}$$

Introducing polar coordinates for $\boldsymbol{\xi}$ gives

$$f(\boldsymbol{x}) = \frac{1}{2\pi} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\boldsymbol{\theta}\cdot\boldsymbol{x}} \hat{f}(\sigma\boldsymbol{\theta}) \sigma d\sigma d\boldsymbol{\theta} \tag{4.32}$$

with $\sigma \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathcal{S}^1$. For ease we conduct the substitution $\gamma = \boldsymbol{\theta} \cdot \boldsymbol{x}$.

$$f(\boldsymbol{x}) = \frac{1}{2\pi} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\gamma} \sigma \hat{f}(\sigma\boldsymbol{\theta}) d\sigma d\boldsymbol{\theta}. \tag{4.33}$$

We then use the Fourier slice theorem inserting $f(\sigma\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \mathscr{F}_r(\mathscr{R}f)(\boldsymbol{\theta}, \sigma)$

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{3}{2}}} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\gamma} \sigma \mathscr{F}_r(\mathscr{R}f)(\boldsymbol{\theta}, \sigma) d\sigma d\boldsymbol{\theta}. \tag{4.34}$$

Using the fact that σ only takes on positive values gives us

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{(2\pi)^2} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\gamma} |\sigma| \mathscr{F}_r(\mathscr{R}f)(\boldsymbol{\theta}, \sigma) d\sigma d\boldsymbol{\theta} \\ &= \frac{1}{(2\pi)^2} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\gamma} |\sigma| \frac{1}{\sqrt{2\pi}} \int_{r\in\mathbb{R}} e^{-i\sigma r} (\mathscr{R}f)(\boldsymbol{\theta}, r) dr d\sigma d\boldsymbol{\theta}. \end{aligned} \tag{4.35}$$

We now conduct the substitutions $\boldsymbol{\theta} \to -\boldsymbol{\theta}$, $\sigma \to -\sigma$ and $r \to -r$, before utilising these properties

$$\sigma\gamma = \sigma\boldsymbol{\theta} \cdot \boldsymbol{x} = (-\sigma)(-\boldsymbol{\theta}) \cdot \boldsymbol{x} \tag{4.36}$$

$$\sigma r = (-\sigma)(-r) \tag{4.37}$$

$$\mathscr{R}f(\boldsymbol{\theta}, r) = \mathscr{R}f(-\boldsymbol{\theta}, -r) \tag{4.38}$$

in order to get

$$\begin{aligned} f(\boldsymbol{x}) =& \frac{1}{(2\pi)^2} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_0^\infty e^{i\sigma\gamma} |\sigma| \frac{1}{\sqrt{2\pi}} \int_{r\in\mathbb{R}} e^{-i\sigma r} (\mathscr{R}f)(\boldsymbol{\theta}, r) dr d\sigma d\boldsymbol{\theta} \\ =& \frac{1}{(2\pi)^2} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_{-\infty}^0 e^{i(-\sigma)(-\gamma)} |-\sigma| \frac{1}{\sqrt{2\pi}} \int_{r\in\mathbb{R}} e^{-i(-\sigma)(-r)} (\mathscr{R}f)(-\boldsymbol{\theta}, -r) dr d\sigma d\boldsymbol{\theta} \\ =& \frac{1}{(2\pi)^2} \int_{\boldsymbol{\theta}\in\mathbb{S}^1} \int_{-\infty}^0 e^{i\sigma\gamma} |\sigma| \frac{1}{\sqrt{2\pi}} \int_{r\in\mathbb{R}} e^{-i\sigma r} (\mathscr{R}f)(\boldsymbol{\theta}, \sigma) dr d\sigma d\boldsymbol{\theta}. \end{aligned} \tag{4.39}$$

Adding (4.39) to (4.35) gives:

$$2f(\boldsymbol{x}) = \frac{1}{(2\pi)^2}\int_{\boldsymbol{\theta}\in\mathbb{S}^1}\int_{-\infty}^{\infty} e^{i\sigma\gamma}|\sigma|\frac{1}{\sqrt{2\pi}}\int_{r\in\mathbb{R}} e^{-i\sigma r}(\mathscr{R}f)(\boldsymbol{\theta},r)dr d\sigma d\boldsymbol{\theta}. \quad (4.40)$$

We notice that $\mathscr{F}_\sigma^{-1} = \frac{1}{2\pi}\int_{\mathbb{R}} e^{i\sigma\gamma}$ where gamma takes the role of the new free variable. This gives

$$2f(\boldsymbol{x}) = \frac{1}{2\pi}\int_{\boldsymbol{\theta}\in\mathbb{S}^1} \mathscr{F}_\sigma^{-1}|\sigma|\mathscr{F}_r(\mathscr{R}f)(\boldsymbol{\theta},\gamma)d\sigma d\boldsymbol{\theta}. \quad (4.41)$$

When we back-substitute $\gamma = \boldsymbol{\theta}\cdot\boldsymbol{x}$ we recognise the backprojection operator and get

$$f(\boldsymbol{x}) = \mathscr{R}^*\mathscr{F}_\sigma^{-1}\frac{|\sigma|}{4\pi}\mathscr{F}_r\mathscr{R}f(\boldsymbol{x}) = \mathscr{R}^*\mathscr{F}^{-1}\frac{|\sigma|}{4\pi}\mathscr{F}g(\boldsymbol{x}). \quad (4.42)$$

□

The name for this algorithm comes comes from the convolution theorem which states that

$$g(x) * f(x) = \mathscr{F}^{-1}\hat{g}(\sigma)(\sigma)\mathscr{F}f(\sigma) \quad (4.43)$$

The filtered backprojection algorithm is in other words equivalent to applying a convolution filter along the r-axis of the sinogram, where the filter has the Fourier transform $\frac{|\sigma|}{4\pi}$. We want to expand on this idea of a filter by the following theorem

Theorem 4.3 (Theorem 2.1.3 from Natterer [10])
Let the function $f \in \mathcal{S}(\mathbb{R}^2)$ and $g \in \mathcal{S}(\mathbb{S}^1\times\mathbb{R})$, we then have

$$\mathscr{R}^*g(\boldsymbol{x}) * f(\boldsymbol{x}) = \mathscr{R}^*(g(\boldsymbol{\theta},r) *_r \mathscr{R}f(\boldsymbol{\theta},r)) \quad (4.44)$$

This means that the convolution filter we use becomes the delta function when backprojected.

4.4 Noise and discretisation

We live in a finite world, and can only collect and manipulate finite amounts of data, measuring the exact Radon transform of a body's attenuation is therefore neither feasible nor ideal, as we don't want to give patients an infinite radiation dose. We therefore need to aproximate the true Radon transform in some shape or form. This is done by sampling the Radon transform at given angles $\boldsymbol{\theta}_i \in \{\theta_0,\theta_1,...,\theta_k\}$ and given positions $r_j \in \{r_{-l},...,r_0,...,r_l\}$. The width of the details we can observe with such a discretisation is given by the famous Nyquist-Shannon sampling theorem.

Definition 4.3 (Band-Limited functions [10])**.** Let the function $f \in \mathcal{S}(\mathbb{R}^n)$, we say that this function is **b-band limited** if

$$b \le ||\boldsymbol{\xi}||_2 \;\Rightarrow\; \mathscr{F}f(\boldsymbol{\xi}) = 0 \quad (4.45)$$

Theorem 4.4 (Shannon-Nyquist Sampling Theorem [10])
Let $f \in L_2(\mathbb{R}^2)$ be b-band limited, $0 < h \leq \frac{\pi}{b}$ and $\boldsymbol{k} \in \mathbb{Z}^2$. The function f is then uniquely defined by:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k}} f(h\boldsymbol{k}) sinc(\boldsymbol{x} - h\boldsymbol{x}) \tag{4.46}$$

We also have that

$$\hat{f}(\boldsymbol{\xi}) = \frac{h^2}{2\pi} \sum_{\boldsymbol{k}} f(h\boldsymbol{k}) e^{-ih\boldsymbol{k}\cdot\boldsymbol{\xi}} \tag{4.47}$$

This theorem tells us something about what sampling frequency we need in order to reconstruct images of a certain detail. The discretisation does in other words mean that the details we observe in the sinogram is dependent on the sampling frequency, and give us a cutoff frequency for what details we can "trust" and not. The next step here is introducing a noise model.

The noise model we will use in this text comes from the book *Medical Imaging Systems* by Albert Macovski[9], which proposes that the noise is Poisson distributed with variance inversely proportional to the measured beam intensity I, $\sigma_I^2 \propto (I)^{-1}$. The biggest weakness of this model is that it assumes that all x-rays have the same intensity. This is not true because the X-ray source produce a spectrum of X-rays with several peaks. It is also important to remember that attenuation is dependent on photon energy as well[14]. These issues will not be considered in this report, as the main focus is regularisation methods, not noise models. We are later going to add noise to our images in order to test different regularisation methods, and are therefore interested in how the noise is related to $\mu(\boldsymbol{x})$. We have, from (4.2), that

$$\mathscr{R}_\theta \mu(s_i) = \int_{L_i} \mu(\boldsymbol{x}) d\boldsymbol{x} = \ln\left(\frac{I_0}{I}\right). \tag{4.48}$$

Combining this with the noise model we use gives

$$\sigma_i^2 = I_0 e^{\mathscr{R}_\theta \mu(s_i)} \tag{4.49}$$

This gives us two new problems, the first being that we need to know the intensity I_0, and the second that we need to know the actual values of our attenuation. The second problem is solved by assuming that the ring in our phantom is 1 cm wide and made of bone, which has attenuation of about $0.5cm^{-1}$[14]. This means that the attenuation is given by the equation

$$\mu(x, y) = \frac{f(x, y)}{2N_p} \tag{4.50}$$

Where N_p is the width of the white ring in our phantom and $f(x, y)$ is the phantom value at coordinates x and y. An illustration of this can be seen in figure 4.5 where we see that the ring has width $4px$ for a 128×128 phantom. Applying this to our integral gives this noise formula

$$\sigma_{i,j}^2 = I_0 e^{\frac{\mathscr{R} f(\boldsymbol{\theta}_j, s_j)}{2N_p}}. \tag{4.51}$$

The thing we are interested in is how big the error in our signal σ_{noise} is as a function of beam intensity I_0.

Proposition 4.4
The noise in attenuation σ_{noise} is given by

$$\sigma_{noise} \approx \frac{1}{\sqrt{I^3}} \tag{4.52}$$

where I_0 is the start intensity of the X-ray beam and I is the measured intensity of the X-ray beam.

Proof. This proof builds on the following rule from the book Introduction to Error Analysis by Taylor [18]

$$\sigma_{noise} \approx \left| \frac{d\mathscr{R}\mu}{dI} \right| \sigma_I. \tag{4.53}$$

Evaluating this derivative with $\mathscr{R}\mu = ln\left(\frac{I_0}{I}\right)$ yields

$$\sigma_{noise} \approx \frac{\sigma_I}{I} = \frac{1}{I\sqrt{I}}. \tag{4.54}$$

□

It is important to note that the rule from Taylor's book [18] uses a Taylor expansion, so this noise model is an approximation, but as we will see later, a good approximation. We are going to conduct some numerical experiments, and need a way of calculating the relative noise as a function of initial beam intensity I_0 and attenuation $\mathscr{R}\mu$.

Proposition 4.5
The relative noise in attenuation $p(\boldsymbol{\theta}, r) = \frac{\sigma_{noise}}{\mathscr{R}\mu(\boldsymbol{\theta},r)}$ is given by

$$p = \frac{1}{\mathscr{R}\mu(\boldsymbol{\theta}, r)\left(I_0 e^{\mathscr{R}\mu(\boldsymbol{\theta},r)}\right)^{\frac{3}{2}}} \tag{4.55}$$

where I_0 is the initial beam intensity and μ is the attenuation. This means that choosing initial beam intensity is done by this equation

$$I_0 = \frac{e^{\mathscr{R}\mu(\boldsymbol{\theta},r)}}{(p\mu)^{\frac{2}{3}}} \tag{4.56}$$

Proof. Proving this comes from the equation given by $\sigma_{noise} = p\mathscr{R}\mu(\boldsymbol{\theta}, r)$:

$$p\mathscr{R}\mu(\boldsymbol{\theta}, r) = \frac{1}{\left(I_0 e^{\mathscr{R}\mu(\boldsymbol{\theta},r)}\right)^{\frac{3}{2}}}. \tag{4.57}$$

The equation is easily solved with respect to p in order to get (4.55) and I_0 in order to get (4.56). □

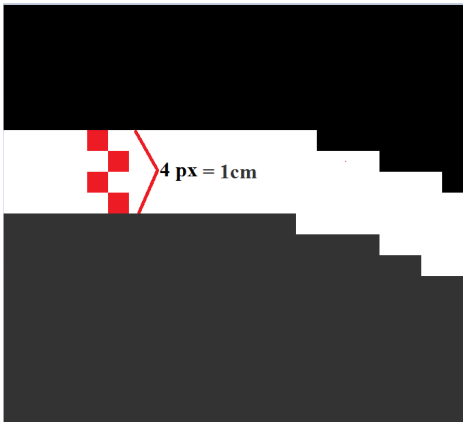

Figure 4.5: Counting number of white pixels in 128×128 phantom shows that the attenuation value in pixel p is phatom value in p divided by 8

4.5 Regularisation of the filtered backprojection

We want to change the way we describe the filtered backprojection method in order to regularise it, and that description comes from Theorem 4.3 which states

$$(\mathscr{R}^* g_\lambda) * f = \Phi_\lambda * f = \mathscr{R}^*(g_\lambda * \mathscr{R} f). \tag{4.58}$$

The regularisation then comes in the form of a family of functions g_λ such that $\mathscr{R}^* g_\lambda = \Phi_\lambda \to \delta$ as $\lambda \to 0$ where δ is the Dirac delta function. We now want to create these functions, a simple way of doing this is simply by introducing a cutoff, and saying that

$$\hat{g}_\lambda(\boldsymbol{\theta}, \sigma) = \begin{cases} 0 & \text{if } \sigma \geq \lambda^{-1} \\ |\sigma| & \text{otherwise} \end{cases} \tag{4.59}$$

This regularisation makes sense for two reasons, the first comes from the sampling theorem and fact that we can't get information that has a higher frequency than $\frac{\pi}{h}$ where h is the sampling frequency. The second reason comes from Theorem 2.3.7 in Natterer [10] that states that the functions in the sampled Radon transform's kernel are highly oscillatory. We can therefore, by limiting the highly oscillatory information of the reconstructed image, lower the chance of reconstructing an image where much of the information comes from the discrete Radon transform's kernel. One might think that using the Nyquist frequency, $\frac{\pi}{h}$, as cutoff frequency is the best choice, but noise is usually highly oscillatory. We might therefore want to remove it by setting the cutoff-frequency lower. This creates both ring-artefacts that look like ripples in water and blurs the image, something that can be seen from Figure 4.6.

Observing these artefacts begs the question of whether or not there exists better methods of reducing noise in the image, while still maintaining its high-

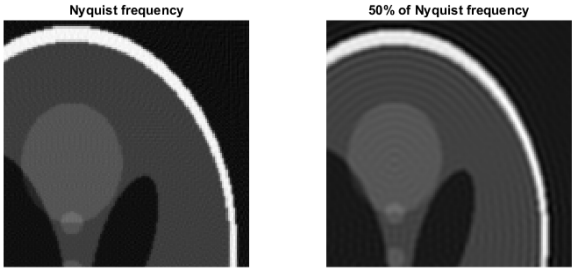

Figure 4.6: To the left: filtered backprojection with no noise and cutoff at nyquist frequency. To the right: filtered backprojection with cutoff at 50% of Nyquist frequency

frequency components. One way of obtaining this is by using smoother a transition than the absolute value filter. An example of such a filter was proposed by Shepp and Logan in 1974 [10] and takes the form

$$\hat{g}_\lambda(\boldsymbol{\theta}, \sigma) = \begin{cases} 0 & \text{if } \sigma \geq \lambda^{-1} \\ |\sigma| \text{sinc}(\frac{\sigma\pi}{2}) & \text{otherwise} \end{cases} \tag{4.60}$$

We can, in order to understand why this filter is better, have look at Figure 4.7. The figure shows that the high-frequency content of the function are weighted less than with a regular absolute value filter. The reasoning for this is the same as with choosing a lower cutoff-frequency than the Nyquist frequency, namely removing the highly oscillating noise. The difference is that we acknowledge that the image also has some highly oscillating details, and don't want to remove them completely. The effect of chosing the Shepp-Logan filter instead of the standard absolute value filter can be seen in Figure 4.8 where a very low cutoff frequency is chosen to exaggerate the effect.

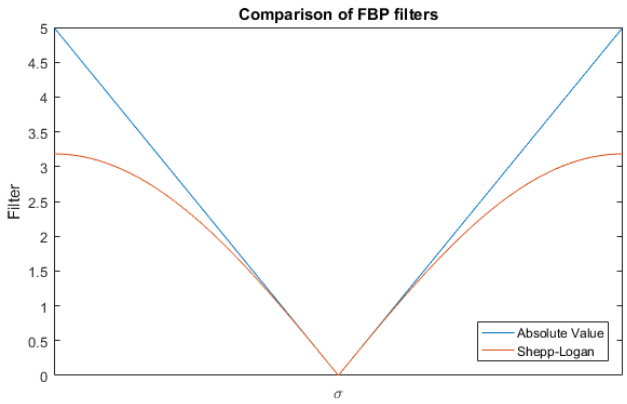

Figure 4.7: Comparison of the Fourier transform of the Shepp-Logan filter and the absolute value filter

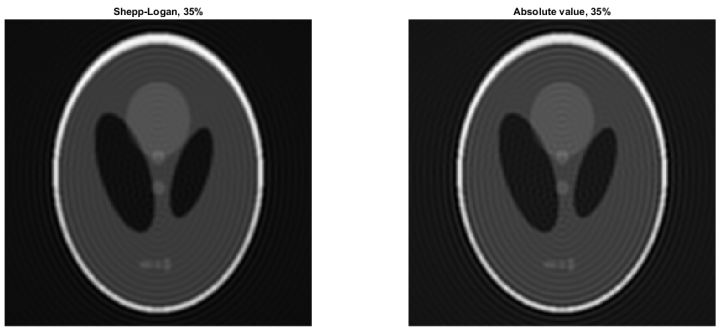

Figure 4.8: Comparison of the Shepp-Logan filter and the absolute value filter. The cutoff-frequency is chosen to be 35% of the Nyquist frequency to clearly demonstrate the difference between the filters.

5 Numerical analysis prerequisites

5.1 Fast Fourier transform

The fast Fourier transform, or FFT for short, is a method of calculating the discrete Fourier tranform (DFT) of a signal in $O(n\log(n))$ operations, instead of the straightforward $O(n^2)$ method. This complexity reduction is obviously important for image reconstruction, as filtered backprojection is a method often used for image reconstruction. The FFT-algorithm is an algorithm with so many applications that it was included in the list of the 10 most important algorithms in the 1900s by the IEEE journal Computing in Science and Engineering [4]. I won't go through the actual algorithms in this text, but rather note one key thing. The FFT algorithm works best if the signal is a power of two and it will, in fact, not have $O(n\log(n))$ complexity if it isn't a power of two. This has one implication for designing software for hospital use, namely that we would add extra 'fictual' observations with zero attenuation at the end of our data. We will in other words be adding fictual streams of photons outside of our patient where no photons are absorbed.

5.2 Gradient-based optimization and FISTA

The most well known algorithm for minimizing a function is probably the Gradient Descent method, a method that uses the gradient to optimize a function with linear convergence. This is however not nearly the best way of optimizing a function using the deriative. The Russian mathematician Yurii Nesterov discovered a first order method with quadratic convergence rate[11]. We will not start with this algorithm, but rather by laying out the gradient descent algorithm, in order to get a better understanding of Nesterov's algorithm.

Algorithm 1 Gradient Descent

Initialise:
  $\boldsymbol{x}_0$: Guess of minimizer
  L: Maximum bound of Lipschitz constant of ∇f
  N: No. of iterations
for $i \leftarrow 1, N$ **do**
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1} - \frac{1}{L}\nabla f(\boldsymbol{x}_{i-1})$
end for
return x_n

This algorithm works by guessing an initial function, then using the gradient to calculate the direction of steepest descent before "taking a step" that

direction. It can be shown that this algorithm has linear convergence. The accelerated gradient algorithm works a bit differently and thus has quadratic convergence:

Algorithm 2 Accelerated gradient optimisation algorithm [3]

Initialise:
$\boldsymbol{x}_0$: Guess of minimizer
$\boldsymbol{y}_1 = \boldsymbol{x}_0$.
$t_1 = 1$
L: Maximum bound of Lipschitz constant of ∇f
N: No. of iterations
for $i \leftarrow 1, N$ **do**
$\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i - \frac{1}{L}\nabla f(\boldsymbol{y}_i)$
$t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i}}{2}$
$\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$
end for
return x_n

The only difference here is that the accelerated doesn't take a gradient step, it takes a longer step that depends on the previous iterations. This gives an improved convergence rate, at the cost of losing the monotonicity of gradient descent, that is $||\boldsymbol{x}_i - \boldsymbol{x}^\star||$ doesn't need to be smaller than $||\boldsymbol{x}_{i-1} - \boldsymbol{x}^\star||$. This problem can be overcome by a restart scheme, as proposed by Brendan O'Donoghue and Emmanuel Candès [12].

The important change O'Donoghue and Candès make is to test, in every iteration, whether or not the objective function is decreased. The algorithm resets the t variable to 1 and sets $\boldsymbol{y}_{i+1} = \boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1}$ if the function hasn't decreased. A way of imagining what this method does is by regarding t as a momentum term, and algorithm builds momentum in the descent direction, this momentum is then regarded when the next step is computed. The restart term will then be equivalent to loosing all momentum[12]. The next difficulty we encounter is finding an estimate for L, and that is unfortunately always possible, and even rarer feasible. So we start with an initial guess for L and increase it if restarting doesn't work in order to get convergence. Every time the objective function increases we test whether or not last iteration was at a restart point. If it was we know the step size $\frac{1}{L}$ is too large, so we need to adjust our guess of Lipschitz constant. This is done until we either reach a maximum guess of Lipschitz constant, or the residual decreases.

There is one big problem with gradient based algorithms, namely that they require the function we are interested in to be differentiable, and the L_1 norm is not differentiable. In order to deal with this we introduce proximal gradient algorithms, algorithms that uses something similar to the gradient step $x_i = x_{i+1} - \frac{1}{L}\nabla f(x_{i+1})$, but that also exists for convex functions with undefined gradient. We start by defining the proximal operator of a function.

Definition 5.1 (Proximal operator)**.** [13] The proximal operator $p_f(\boldsymbol{x})$ of the

Algorithm 3 Accelerated gradient optimisation algorithm with restart scheme [3][12]

Initialise:
  $\boldsymbol{x}_0$: Guess of minimizer
  $\boldsymbol{y}_1 = \boldsymbol{x}_0$.
  $t_1 = 1$
  L: Maximum bound of Lipschitz constant of ∇f
  N: No. of iterations
for $i \leftarrow 1, N$ **do**
  $\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i - \frac{1}{L}\nabla f(\boldsymbol{y}_i)$
  if $f(\boldsymbol{x}_i) \leq f(\boldsymbol{x}_{i-1})$ **then**
    $t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i}}{2}$
    $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$
  else
    $t_{i+1} \leftarrow 1$
    $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_{i-1}$
  end if
end for
return x_n

convex function $f : \mathbb{R}^n \to \mathbb{R}$ is given by:

$$p_f(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\{f(\boldsymbol{y}) + \frac{1}{2}||\boldsymbol{y} - \boldsymbol{x}||_2^2\}. \tag{5.1}$$

Proposition 5.1
Let $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ be a convex function with minimizer $\boldsymbol{x}^\star$, $\boldsymbol{x}^\star$ is then a fixed point for the proximal map $\boldsymbol{x}_{k+1} = p_f(\boldsymbol{x}_k)$.

Proof. $\boldsymbol{x}^\star$ is a fixed point of the map $\boldsymbol{x}_{k+1} = p(\boldsymbol{x}_k)$ if and only if $p_f(\boldsymbol{x}^\star) = \boldsymbol{x}^\star$. We also know that $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and that the norm $||\boldsymbol{y}|| \geq 0||$ with $||\boldsymbol{y}|| = 0||$ if and only if $\boldsymbol{y} = 0$. Proving proposition 5.1 can therefore easily be done by calculating $p_f(\boldsymbol{x}^\star)$

$$p_f(\boldsymbol{x}^\star) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\left\{f(\boldsymbol{y}) + \frac{1}{2}\,||\boldsymbol{y} - \boldsymbol{x}^\star||\right\}. \tag{5.2}$$

We then test if $\boldsymbol{x}^\star$ is a minimizer of that expression

$$f(\boldsymbol{x}^\star) + \frac{1}{2}\,||\boldsymbol{x}^\star - \boldsymbol{x}^\star|| = f(\boldsymbol{x}^\star) + \frac{1}{2}\,||0|| \tag{5.3}$$

We know that $||0||$ is the smallest possible norm and that $f(\boldsymbol{x}^\star)$ is the smallest possible f-value. This means that $y = \boldsymbol{x}^\star$ has to be the minimiser of $f(\boldsymbol{y}) + \frac{1}{2}\,||\boldsymbol{y} - \boldsymbol{x}^\star||$ which concludes our poof. □

Algorithm 4 Accelerated gradient optimisation algorithm with backtracking and restart scheme [3][12]

Initialise:
$\boldsymbol{x}_0$: Guess of minimizer
$\boldsymbol{y}_1 = \boldsymbol{x}_0$.
$t_1 = 1$
L: Initial guess of Lipschitz constant of ∇f
L_{max}: Maximum acceptable guess for Lipschitz constant
$q \in (0, 1)$: Lipschitz guess increase parameter
N: No. of iterations
for $i \leftarrow 1, N$ **do**
  $\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i - \frac{1}{L}\nabla f(\boldsymbol{y}_i)$
  if $f(\boldsymbol{x}_i) \leq f(\boldsymbol{x}_{i-1})$ **then**
    $t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i}}{2}$
    $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$
  else
    if $t_i \neq 1$ **then** ▷ Restarted last iteration if $t_i = 1$
      $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$
      $t_{i+1} \leftarrow 1$
      $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_{i-1}$
    else
      while $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i - \frac{1}{L}\nabla f(\boldsymbol{y}_i))$ **do**
        $L \leftarrow (1 + q)L$
        $t_{i+1} \leftarrow 1$
        if $L \geq L_{max}$ **then**
          break for and while
        end if
      end while
      $\boldsymbol{x}_i \leftarrow \boldsymbol{y}_i - \frac{1}{L}\nabla f(\boldsymbol{y}_i)$
      $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i$
    end if
  end if
end for
return x_i

It is also possible to show that the proximal map has sub-linear convergence toward this fixed point[1]. Sublinear convergence is not ideal, but it is, luckily, a way to to obtain quadratic convergence for problems on the form:

$$\arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \{f(\boldsymbol{x}) + g(\boldsymbol{x})\} \tag{5.4}$$

with $f(x) \in C^1(\mathbb{R}^2)$ and $g(x) \notin C^1(\mathbb{R}^n)$. This is done using *proximal gradient methods*.

Definition 5.2. The operator $P_F(\boldsymbol{x}; L)$, which we in this text will call the modified proximal operator, of a convex function $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ with $f(x) \in C^1(\mathbb{R}^n)$ and $g(x) \notin C^1(\mathbb{R}^n)$ is given by

$$P_F(\boldsymbol{x}; L) = p_{\frac{1}{L}g}\left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right) = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\left\{g(\boldsymbol{y}) + \frac{L}{2}\left|\left|\boldsymbol{y} - \left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right)\right|\right|_2^2\right\} \tag{5.5}$$

To better understand this operator we have a look at what corresponds to when $F(\boldsymbol{x}) = f(\boldsymbol{x})$ and when $F(\boldsymbol{x}) = g(\boldsymbol{x})$.

Proposition 5.2
Modified proximal operator corresponds to a gradient descent step in $\boldsymbol{x}$ for a function $F(\boldsymbol{x}) = f(\boldsymbol{x})$ with $f(\boldsymbol{x}) \in C(\mathbb{R}^n)$ and f convex. The modified proximal operator corresponds to applying the ordinary proximal operator (5.1) *in $\boldsymbol{x}$ on $\frac{1}{L}g(\boldsymbol{x})$ if applied to the function $F(\boldsymbol{x}) = g(\boldsymbol{x})$ with $f(\boldsymbol{x}) \notin C(\mathbb{R}^n)$ and g convex.*

Proof. The second part of this proposition is very easy to prove, just set $\nabla f(\boldsymbol{x}) = 0$ in (5.5) and get

$$P_F(\boldsymbol{x}; L) = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\left\{g(\boldsymbol{y}) + \frac{L}{2}\left|\left|\boldsymbol{y} - \boldsymbol{x}\right|\right|_2^2\right\} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\left\{\frac{1}{L}g(\boldsymbol{y}) + \frac{1}{2}\left|\left|\boldsymbol{y} - \boldsymbol{x}\right|\right|_2^2\right\}. \tag{5.6}$$

The last bit of this comes from the fact that the minimiser of $h(\boldsymbol{x})$ is the same as the minimiser of $Lh(\boldsymbol{x})$. The next part of this proof is a bit trickier, but still quite doable, and comes solving (5.5) with $g(\boldsymbol{x}) = 0$.

$$P_F(\boldsymbol{x}; L) = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\frac{L}{2}\left|\left|\boldsymbol{y} - \left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right)\right|\right| \tag{5.7}$$

expanding the norm and removing the constant L from outside it yields

$$\begin{aligned} P_F(\boldsymbol{x}; L) &= \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\frac{1}{2}\left(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T\left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right) + \left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right)^T\left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right)\right) \\ &= \arg\min_{\boldsymbol{y}\in\mathbb{R}^n} h(\boldsymbol{y}). \end{aligned} \tag{5.8}$$

We know that the minimiser of this is the value $\boldsymbol{y}$ such that the gradient $\nabla_y h(\boldsymbol{y})$ is equal to zero,

$$\nabla_y h(\boldsymbol{y}) = \frac{1}{2}\left(2\boldsymbol{y} - 2\left(\boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x})\right)\right) = 0. \tag{5.9}$$

Solving this equation gives

$$\boldsymbol{y} = \boldsymbol{x} - \frac{1}{L}\nabla_x f(\boldsymbol{x}) \tag{5.10}$$

which is the normal gradient step with steplength $\frac{1}{L}$. □

This proposition gives merit to using the modified proximal operator, as is an attempt to use as much information from the differentiable parts of F as possible. Algorithm 5 is our first attempt at a way to generate an algorithm to minimise functions with a continuously differentiable part and a continuous nondifferentiable part. This algorithm has linear convergence, but Beck

Algorithm 5 Simple proximal gradient algorithm

[13]

Initialise:
$F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$
$f \in C^1(\mathbb{R}^n)$ and $g \notin C^1(\mathbb{R}^n)$
$\boldsymbol{x}_0$: Guess of minimizer
L: Maximum bound of Lipschitz constant of ∇f
N: No. of iterations
for $i \leftarrow 1, N$ **do**
$\boldsymbol{x}_i \leftarrow P_F(\boldsymbol{x}_{i-1}; L)$
end for
return x_n

and Teboulle proved methods using this modified proximal operator can get quadratic convergence rate combining it with Nesterov's accelerated scheme[3]. They called their algorithm FISTA, for Fast Iterative Shrinkage-Thresholding Algorithm.

We can improve this algorithm by using O'Donoghue and Candès' restart scheme [12] and backtracking for the Lipschitz constant, resulting in this algorithm.

This is the algorithm used in all numerical experiments later in the text.

5.3 Calculating singular values

We will later be interested in finding the largest singular value of a matrix, something that can be done with the power method. Algorithm 8 works very well if there are large differences between the largest and second largest singular values, and the smaller those differences are, the slower it converges.

Algorithm 6 FISTA [3]

Initialise:
$F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$
$f \in C^1(\mathbb{R}^n)$ and $g \notin C^1(\mathbb{R}^n)$
$\boldsymbol{x}_0$: Guess of minimizer
$\boldsymbol{y}_1 = \boldsymbol{x}_0$.
$t_1 = 1$
L: Maximum bound of Lipschitz constant of ∇f
N: No. of iterations
for $i \leftarrow 1, N$ **do**
$\boldsymbol{x}_i \leftarrow P_F(\boldsymbol{y}_i; L)$
$t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i}}{2}$
$\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$
end for
return x_n

Algorithm 7 FISTA with backtracking and restart scheme [3][12]

Initialise:
$F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$
$f \in C^1(\mathbb{R}^n)$ and $g \notin C^1(\mathbb{R}^n)$
$\boldsymbol{x}_0$: Guess of minimizer
$\boldsymbol{y}_1 = \boldsymbol{x}_0$.
$t_1 = 1$
L: Initial guess of Lipschitz constant of ∇f
L_{max}: Maximum acceptable guess for Lipschitz constant
$q \in (0, 1)$: Lipschitz guess increase parameter
N: No. of iterations
for $i \leftarrow 1, N$ **do**
  $\boldsymbol{x}_i \leftarrow P_F(y_i; L)$
  if $F(\boldsymbol{x}_i) \leq F(\boldsymbol{x}_{i-1})$ **then**
    $t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i}}{2}$
    $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$
  else
    if $t_i \neq 1$ **then** ▷ Restarted last iteration if $t_i = 1$
      $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$
      $t_{i+1} \leftarrow 1$
      $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_{i-1}$
    else
      while $F(\boldsymbol{x}_i) < F(P_F(\boldsymbol{y}_{i-1}; L))$ **do**
        $L \leftarrow (1 + q)L$
        $t_{i+1} \leftarrow 1$
        if $L \geq L_{max}$ **then**
          break for and while
        end if
      end while
      $\boldsymbol{x}_i \leftarrow P_L(y_i; L)$
      $\boldsymbol{y}_{i+1} \leftarrow \boldsymbol{x}_i$
    end if
  end if
end for
return x_i

Algorithm 8 Power method for computing singular values, modified from [8]

Initialise:
$A \in \mathbb{C}^{n \times m}$
$\boldsymbol{v}_0 = \boldsymbol{v}_1 \neq 0 \in \mathbb{R}^m$
$\epsilon > 0$
$\sigma_1 \leftarrow 0$
$i \leftarrow 1$
while $||\boldsymbol{v}_i||_2 - ||\boldsymbol{v}_{i-1}||_2 > \epsilon or i < N$ **do**
$\boldsymbol{v}_{i+1} \leftarrow A^* A \boldsymbol{v}_{i+1}$
$\sigma_{i+1}^2 \leftarrow ||\boldsymbol{v}_{i+1}||$
$\boldsymbol{v}_{i+1} \leftarrow \frac{\boldsymbol{v}_{i+1}}{\sigma_{i+1}^2}$
$i \leftarrow i + 1$
end while
return σ_i

6 Numerical experiments

All experiments are done with noisy data, with relative noise of 5% and 10%, attained by using proposition 4.5 wit h the average sinogram value. The discrete Radon transform was conducted with the built-in MATLAB function `radon(P, thetas)` where `P` is the image to transform and `thetas` are the projection angles. Backprojection was done with `iradon(R, thetas, 'linear', 'none')` where `R` is sinogram returned by the `radon` function and `thetas` are the projection angles.

There are two problems with the built in MATLAB discretisations, the first is that the `iradon` function is not the true adjoint of the discrete Radon transform, but rather a discrete approximation of the backprojection transformation. We experience this as a small, or possibly no, error so our iterative algorithms still works fine. The second problem is that the `radon` function returns the same size sinogram for multiple image sizes. This means that the image size can change after backprojection. We have, by trial and error, found out that the right crop for images with dimensions equal to a power of 2 is by skipping the first and last row and column. MATLAB also has a filtered backprojection algorithm built in, and that is the one we will use comparing the results from our iterative reconstructions as it performs better than our filtered backprojection algorithm. The next obstacle we faced was finding the right regularisation parameters, which was done heuristically.

6.1 Residuals

We will also require some vocabulary for the next couple of sections. We will call $r(\boldsymbol{x};\boldsymbol{g}) = ||\mathscr{R}_D\boldsymbol{x}-\boldsymbol{g}||_2^2$ the **residual** for test image $\boldsymbol{x}$ and observed sinogram $\boldsymbol{g}$. The gradient of this residual is interesting for all the iterative reconstruction algorithms used in this text.

Proposition 6.1
The graident $\nabla r(f_i; g)$ is given by

$$\nabla r = 2\mathscr{R}_D^*(\mathscr{R}\boldsymbol{x} - \boldsymbol{g}) \tag{6.1}$$

Where $\mathscr{R}_D$ is the discrete approximation of the Radon transform.

Proof. We know, from linear algebra, that any linear transformation between finite-dimensional vector spaces can be written as a matrix transformation in $\mathbb{R}^n$. Using this and the norm $||\boldsymbol{y}|| = \boldsymbol{y}\cdot\boldsymbol{y}$

$$\nabla r = ||\mathscr{R}_D\boldsymbol{x} - \boldsymbol{g}||_2^2 = \nabla\left((\mathscr{R}_D\boldsymbol{x})^*\mathscr{R}_D\boldsymbol{x} - 2(\mathscr{R}_D\boldsymbol{x})^*\boldsymbol{g} + \boldsymbol{g}^*\boldsymbol{g}\right). \tag{6.2}$$

Evaluating the ∇ operator yields

$$\nabla r = 2\mathscr{R}_D^* \boldsymbol{x} - 2\mathscr{R}_D \boldsymbol{g} = 2\mathscr{R}_D^*(\mathscr{R}_D \boldsymbol{x} - \boldsymbol{g}) \tag{6.3}$$

which proves (6.1). □

6.1.1 Lipschitz constant for the residual gradient

Knowing the Lipschitz constant of ∇r is useful in order to know what to use as our maximum step size in our iterations. Calculating it is easily done numerically, using the following theorem.

Theorem 6.1
The Lipschitz constant $L_{\nabla r}$ of ∇r is given by $\sqrt{2}\sigma^2$, where $\sigma 2$ is the largest singular value of $\mathscr{R}_D^*\mathscr{R}_D$.

Proof. A Lipschitz constant $L_{\nabla r}$ of ∇r is any constant such that

$$\frac{||\nabla r(\boldsymbol{x}) - \nabla r(\boldsymbol{y})||}{||\boldsymbol{x} - \boldsymbol{y}||} < L_{\nabla r} \tag{6.4}$$

for any $\boldsymbol{x}$ and $\boldsymbol{y}$ in the domain $\mathcal{D}(\mathscr{R}_D^*\mathscr{R}_D)$. Writing the expression for ∇r gives

$$\frac{||2\mathscr{R}_D^*(\mathscr{R}_D \boldsymbol{x} - \boldsymbol{g}) - 2\mathscr{R}_D^*(\mathscr{R}_D \boldsymbol{y} - \boldsymbol{g})||}{||\boldsymbol{x} - \boldsymbol{y}||} < L_{\nabla r}. \tag{6.5}$$

Manipulating this gives

$$\sqrt{2}\frac{||\mathscr{R}_D^*\mathscr{R}_D(\boldsymbol{x} - \boldsymbol{y})||}{||\boldsymbol{x} - \boldsymbol{y}||} < L_{\nabla r}. \tag{6.6}$$

Using the definition of singular values gives us this equation

$$\sqrt{2}\frac{||\mathscr{R}_D^*\mathscr{R}_D(\boldsymbol{x} - \boldsymbol{y})||}{||\boldsymbol{x} - \boldsymbol{y}||} < \sqrt{2}\sigma^2\frac{||\boldsymbol{x} - \boldsymbol{y}||}{||\boldsymbol{x} - \boldsymbol{y}||} = \sqrt{2}\sigma^2 \tag{6.7}$$

Which means that $\sqrt{2}\sigma^2$ satisfies the condition of a Lipschitz constant for ∇r □

Finding the largest singular values to $\mathscr{R}_D^*\mathscr{R}$ is easily done in MATLAB, using the power method. Our calculations resulted in $\sigma^2 \approx 194$, which gives

$$L_{\nabla r} \approx 275. \tag{6.8}$$

6.2 Noise comparison

One might ask if there is a big difference in choosing the noise model derived in this text and just normally distributed with standard deviation given by

$$\sigma_{i,j} = p\mathscr{R}\mu(\boldsymbol{\theta}_i, s_j). \tag{6.9}$$

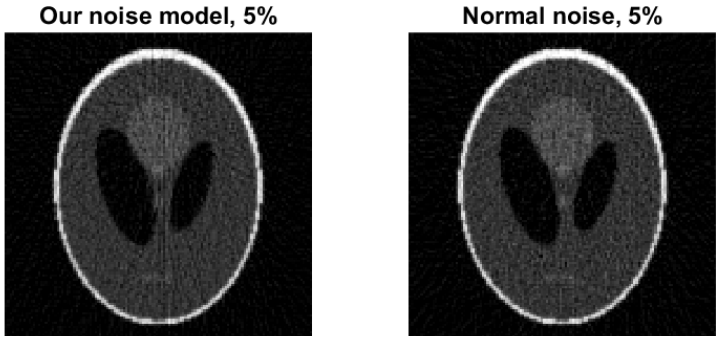

Figure 6.1: Comparison of the noise model we use, to the left, and normally distributed noise, to the right, at a 5% relative noise level

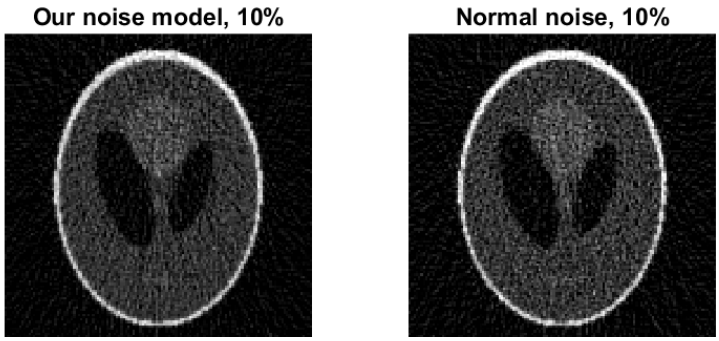

Figure 6.2: Comparison of the noise model we use and normally distributed noise at high noise levels

Testing this is easily done by doing numerical experiments, and that is, indeed, what we have done. We have conducted experiments with relative noise at 5% and 10% in order to see how the noise differs at the level in which we conduct our experiments. The results from the comparison at 20% and 40% is also included in order to get a stronger conclusion about our noise model. We see, in figure 6.1, that the noise model we use gives more "structured" noise, it can almost look as if the noise creates lines in our sinogram. Whereas the normally distributed noise looks completely unstructured. It is also interesting to see that we can see the same details in both pictures. The observations from figure 6.1 continue in 6.2, where we also see the same amount of details, but once again, the our noise model seems to give more "structured" noise than the normally distributed noise.

It is in figure 6.3 things really start to be interesting, the different noise models generate big qualtiative differences between images. Our model has a generally lower pixel intensity than the normally distributed noise, and the normally distributed noise seem to reduce contrasts more than our noise model,

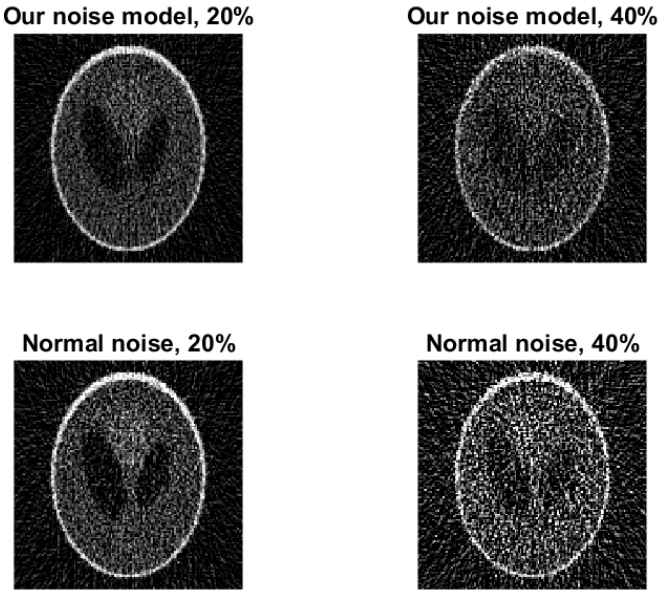

Figure 6.3: Comparison of the noise model we use and normally distributed noise at high noise levels

even though we can make out the same amount of details with both models.

6.3 Filtered backprojection with noise

Filtered back projection is used quite a lot in image reconstruction, so much that MATLAB has a built in filtered back projection function. There is unfortunately no way to specify cutoff frequency in this built-in function so I had to construct a custom function to test out the effect of cutoff frequencies.

The code we wrote for filtered backprojection has been used eariler in the text to show ring artefacts and the difference between the Shepp-Logan filter and just normal absolute value with cutoff. We will therefore only conduct experiments with the Shepp-Logan filter.

The code spent under 3 seconds to compute on my 3-year-old laptop and the results seems quite good. There were to my surprise no ring artefacts in our reconstruction, but that might be because our image doesn't have high enough frequency changes. It is also interesting to see that choosing lower cutoff frequency wasn't a very effective regularisation method in our experiments. It is also important to note that the three small details on the bottom of the phantom are completely lost at 10% noise.

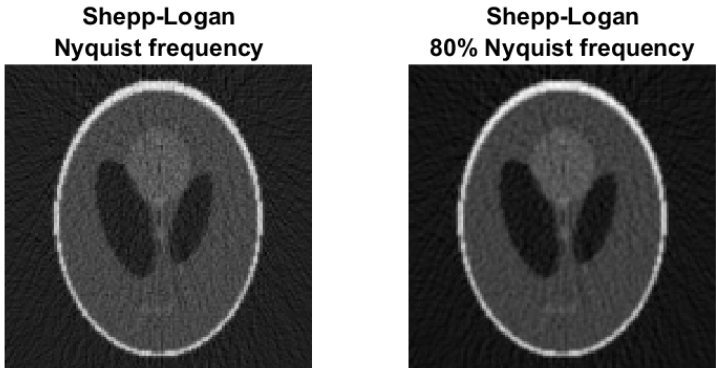

Figure 6.4: Both images are reconstructions with 5% noise. To the left: Cutoff at the Nyquist frequency. To the right: Cutoff at 80% of the Nyquist frequency.

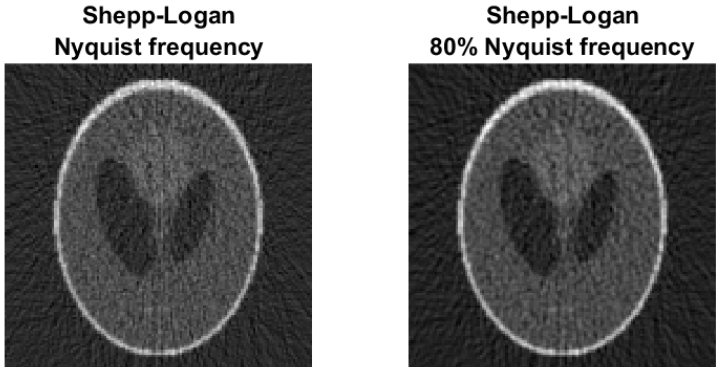

Figure 6.5: Both images are reconstructions with 10% noise. To the left: Cutoff at the Nyquist frequency. To the right: Cutoff at 80% of the Nyquist frequency.

6.4 Tikhonov regularisation

Tikhonov regularisation is conducted with the FISTA algorithm. We know from eariler that we want to compute

$$\arg\min_f ||\mathscr{R}f - g||_2^2 + \lambda||f||_2^2 = \arg\min_{\boldsymbol{x}} r(\boldsymbol{x}) + \lambda||\boldsymbol{x}||_2^2 \tag{6.10}$$

The gradient of this is given by

$$\nabla(\arg\min_{\boldsymbol{x}} r(\boldsymbol{x}) + \lambda||\boldsymbol{x}||_2^2) = 2(\mathscr{R}^*(\mathscr{R}\boldsymbol{x} + g) + \boldsymbol{x}). \tag{6.11}$$

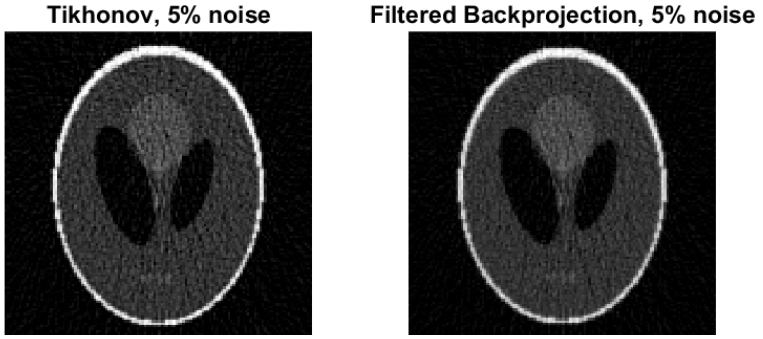

Figure 6.6: Both images are reconstructions with 5% noise. To the left: Tikhonov regularisation with $\lambda = 0.25$. To the right: Filtered Back Projection with Shepp-Logan filter

The proximal gradient operator $P_{Tikhonov}(\boldsymbol{x}; L)$ is then given by

$$P_{Tikhonov}(\boldsymbol{x}; L) = \boldsymbol{x} - \frac{2}{L}(\mathscr{R}^*(\mathscr{R}\boldsymbol{x} + g) + \boldsymbol{x}). \tag{6.12}$$

This gives us an easy way of reconstructing an image using our optimised FISTA algorithm with a maximum of 100 iterations. These results are given in figure 6.6 and 6.7

Tikhonov regularisation seems to give a worse result than just normal filtered backprojection, both with 5 and 10% noise. Not only because of the tiny bit more noise in the 5% case, and the blur in the 10% case, but also because it took over a minute to run.

6.5 L_1 regularisation

L_1 regularisation is a bit more tricky than normal Tikhonov regularisation because the L_1 term isn't continuously differentiable. The target function we want to minimize is given by

$$\arg\min_{\boldsymbol{x}} r(\boldsymbol{x}) + \lambda||\boldsymbol{x}||_1 \tag{6.13}$$

We then need to calculate the proximal gradient of this function

Proposition 6.2
The proximal gradient, P_{L1}, of the L_1 optimization problem is given by the vector with elements

$$[P_{L1}]_i(\boldsymbol{x}) = \begin{cases} x_i - \frac{1}{L}\nabla[r(\boldsymbol{x})]_i + \frac{\lambda}{L} & if\ Lx_i - \nabla[r(\boldsymbol{x})]_i < \lambda \\ x_i - \frac{1}{L}\nabla[r(\boldsymbol{x})]_i - \frac{\lambda}{L} & if\ Lx_i - \nabla[r(\boldsymbol{x})]_i > \lambda \\ 0 & otherwise \end{cases} \tag{6.14}$$

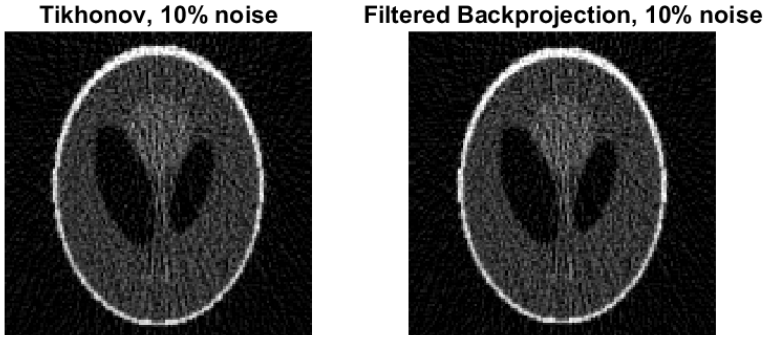

Figure 6.7: Both images are reconstructions with 10% noise. To the left: Tikhonov regularisation with $\lambda = 0.75$. To the right: Filtered Back Projection with Shepp-Logan filter

Proof. We start off with the definition of our modified proximal map

$$P_f(\boldsymbol{x}, L) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\, g(\boldsymbol{y}) + \frac{L}{2}\left|\left|\boldsymbol{y} - \left(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x})\right)\right|\right|_2^2. \tag{6.15}$$

The next step is conducting the substitution $\boldsymbol{z} = \left(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x})\right)$ and inserting for $g(\boldsymbol{y})$ and $||\boldsymbol{y} - \boldsymbol{z}||$.

$$P_f(\boldsymbol{x}, L) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\, \frac{\lambda}{L}||\boldsymbol{y}||_1 + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{z} + \frac{1}{2}\boldsymbol{z}^T\boldsymbol{z}. \tag{6.16}$$

We can then remove the $\boldsymbol{z}^T\boldsymbol{z}$ term since it doesn't vary with $\boldsymbol{y}$ and expand the dot products and the one norm to get

$$P_f(\boldsymbol{x}, L) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\, \frac{\lambda}{L}\sum_{i=1}^{n}|y_i| + \frac{1}{2}\sum_{i=1}^{n}y_i^2 + \sum_{i=1}^{n}y_i z_i. \tag{6.17}$$

These sums don't have any cross terms, and can therefore be combined

$$P_f(\boldsymbol{x}, L) = \underset{\boldsymbol{y}\in\mathbb{R}^n}{\arg\min}\, \frac{\lambda}{L}\sum_{i=1}^{n}\left\{|y_i| + \frac{1}{2}y_i^2 + y_i z_i\right\}. \tag{6.18}$$

Minimising this sum corresponds to minimising each term on its own, so the minimisation problems becomes

$$\underset{y_i\in\mathbb{R}}{\arg\min}\, \frac{\lambda}{L}\left\{|y_i| + \frac{1}{2}y_i^2 + y_i z_i\right\} = \underset{y_i\in\mathbb{R}}{\arg\min}\, h(y_i). \tag{6.19}$$

We then differentiate the function $h(y_i)$ with respect to y_i

$$h'(y_i) = \begin{cases} \frac{\lambda}{L} + y_i - z_i & if y_i > 0 \\ -\frac{\lambda}{L} + y_i - z_i & if y_i < 0 \end{cases} \tag{6.20}$$

Setting this equal to zero yields

$$y_i = \begin{cases} z_i - \frac{\lambda}{L} & if y_i > 0 \Leftrightarrow z_i > \frac{\lambda}{L} \\ z_i + \frac{\lambda}{L} & if y_i < 0 \Leftrightarrow z_i < -\frac{\lambda}{L} \end{cases} \tag{6.21}$$

We know, since h is convex, that h only has one minima and no maxima so (6.21) has to be minima of h. The next step is testing whether or not 0 minimises the equation for $z_i \in \left[-\frac{\lambda}{L}, \frac{\lambda}{L}\right]$. This is true if both the left and right slope of $h(y_i)$ is positive, since that means that h increases in both directions.

$$\lim_{y_i \to 0^+} h'(y_i) = \frac{\lambda}{L} - z_i \quad \lim_{y_i \to 0^-} h'(y_i) \qquad = -\frac{\lambda}{L} - z_i \tag{6.22}$$

We see that the right slope, $\lim_{y_i \to 0^+} h'(y_i)$, is positive if and only if $z_i < \frac{\lambda}{L}$, and the left slope, $\lim_{y_i \to 0^-} h'(y_i)$, is positive if and only if $z_i > -\frac{\lambda}{L}$. We have therefore shown that the minimiser of h is given by

$$y_i = \begin{cases} z_i - \frac{\lambda}{L} & if y_i > 0 \Leftrightarrow z_i > \frac{\lambda}{L} \\ z_i + \frac{\lambda}{L} & if y_i < 0 \Leftrightarrow z_i < -\frac{\lambda}{L} \\ 0 & otherwise \end{cases} \tag{6.23}$$

□

Once we have this proximal map we can use it with the FISTA algorithm, which once again takes about a minute to compute. But it also resulted in quite noisy images as we can see in figure 6.8 and 6.9. These images show exactly what we expect, namely a sparse solution which means many black pixels. This does, unfortunately, also induce more noise on the actual phantom (excluding the black bits around) than the normal backprojection algorithm.

6.6 TV-Seminorm regularisation

The total variation regularisation is by far the most difficult optimisation problem, but luckily Beck and Teboulle published a way of efficiently solving this problem, in 2009 [2]. This method works both for the isotropic and anisotropic discretisation, and we will conduct these experiments with the anisotropic version. The way it is solved is by generating a smooth dual problem to the TV denoising problem

Definition 6.1 (TV denoising dual problem and notation[2])**.** Given the TV denoising problem

$$Dc(\boldsymbol{x}, \lambda) = \arg\min_{\boldsymbol{y}} ||\boldsymbol{y} - \boldsymbol{x}_0||_2^2 + \lambda ||\boldsymbol{y}||_{TV} \tag{6.24}$$

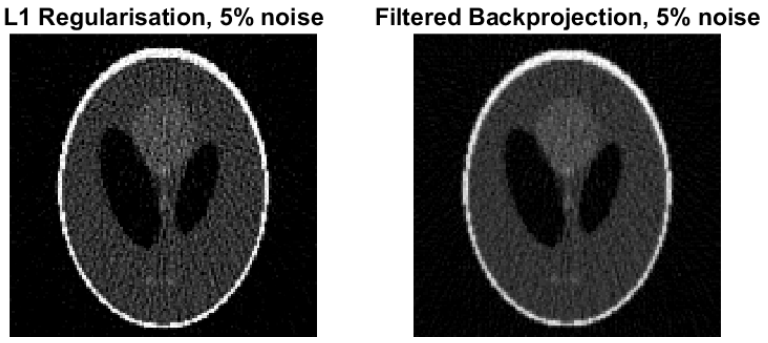

Figure 6.8: Both images are reconstructions with 5% noise. To the left: L_1 regularisation with $\lambda = 0.05$. To the right: Filtered Back Projection with Shepp-Logan filter

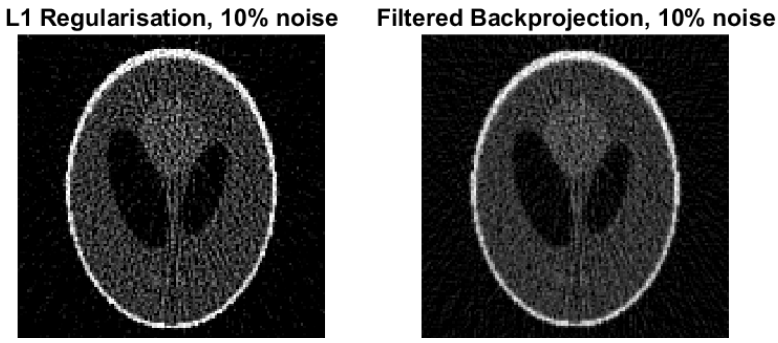

Figure 6.9: Both images are reconstructions with 10% noise. To the left: L_1 regularisation with $\lambda = 0.10$. To the right: Filtered Back Projection with Shepp-Logan filter

with $x \in \mathbb{R}^{m\times n}$ and λ positive. It's dual problem regards a set $\mathcal{P}$ of paired matrices $(\boldsymbol{p}, \boldsymbol{q})$ with the property that $\boldsymbol{p} \in \mathbb{R}^{(m-1)\times n}$ and $\boldsymbol{q} \in \mathbb{R}^{m\times(n-1)}$. The entries of $p_{i,j}$ and $q_{i,j}$ also has to satisfy

$$p_{i,j}^2 + q_{i,j}^2 < 1 \qquad i \in \mathbb{Z}_{m-1} and j \in \mathbb{Z}_{n-1} \tag{6.25}$$

$$|p_{i,m}| < 1 \qquad i \in \mathbb{Z}_{m-1} \tag{6.26}$$

$$|q_{n,j}| < 1 \qquad j \in \mathbb{Z}_{n-1}. \tag{6.27}$$

We also have the linear operator $\mathcal{L} : \mathbb{R}^{(m-1)\times n} \times \mathbb{R}^{m\times(n-1)} :\to \mathbb{R}^{m\times n}$ given by

$$[\mathcal{L}(\boldsymbol{p}, \boldsymbol{q})]_{i,j} = p_{i,j} - p_{i-1,j} + q_{i,j} - q_{i,j-1} \tag{6.28}$$

We assume that $p_{0,j} = p_{m,j} = q_{i,0} = q_{i,m} = 0$ for every $i \in \mathbb{Z}_m$ and $j \in \mathbb{Z}_n$. The transpose of this linear operator is given by

$$\mathcal{L}^T(\boldsymbol{x}) = (\boldsymbol{p}, \boldsymbol{q}) \in \mathbb{R}^{m-1\times n} \times \mathbb{R}^{m,n-1} \tag{6.29}$$

where

$$p_{i,j} = x_{i,j} - x_{i+1,j}, \qquad i \in \mathbb{Z}_{m-1}, j \in \mathbb{Z}_n \tag{6.30}$$

$$q_{i.j} = x_{i,j} - x_{i,j+1}, \qquad i \in \mathbb{Z}_m, j \in \mathbb{Z}_{n-1}. \tag{6.31}$$

We will also need the orthogonal projection operator Π_S on the set S. This gives the operator

$$\Pi_{[l,u]^{m\times n}} = \begin{cases} l & , x_{i,j} < l \\ x_{i,j} & , x_{i,j} \in [l, u] \\ u & , x_{i,j} > u \end{cases} \tag{6.32}$$

If $S = [l, u]^{m\times n}$, and

$$\Pi_{\mathcal{P}}(\boldsymbol{p}, \boldsymbol{q}) = (\boldsymbol{r}, \boldsymbol{s}) \tag{6.33}$$

where the entries of the matrices $\boldsymbol{r}$ and $\boldsymbol{s}$ is given by

$$r_{i,j} = \begin{cases} \frac{p_{i,j}}{\max\{1, \sqrt{p_{i,j}^2 + q_{i,j}^2}\}} & i \in \mathbb{Z}_{m-1} \text{ and } j \in \mathbb{Z}_{m-1} \\ \frac{p_{i,n}}{\max\{1, |p_{i,n}\}|} & \in \mathbb{Z}_{m-1} \end{cases} \tag{6.34}$$

and

$$s_{i,j} = \begin{cases} \frac{q_{i,j}}{\max\{1, \sqrt{p_{i,j}^2 + q_{i,j}^2}\}} & i \in \mathbb{Z}_{m-1} \text{and} j \in \mathbb{Z}_{m-1} \\ \frac{q_{m,j}}{\max\{1, |p_{m,j}|\}} & j \in \mathbb{Z}_{n-1}. \end{cases} \tag{6.35}$$

Using this notation gives the dual problem

Theorem 6.2 (TV regularisation dual problem [2])
The optimal solution to(6.24) *is given by*

$$\boldsymbol{x} = \Pi_{[l,u]^{n\times m}}(\boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}(\boldsymbol{p}, \boldsymbol{q})) \tag{6.36}$$

where $(\boldsymbol{p}, \boldsymbol{q})$ *is given by*

$$(\boldsymbol{p}, \boldsymbol{q}) = \underset{(\boldsymbol{p},\boldsymbol{q})\in\mathcal{P}}{\arg\min}\, h(\boldsymbol{p}, \boldsymbol{q})$$
$$\underset{(\boldsymbol{p},\boldsymbol{q})\in\mathcal{P}}{\arg\min} \left\{ \left|\left| \boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}\left(\boldsymbol{p}, \boldsymbol{q}\right) \right|\right|_F - \left|\left| \boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}(\boldsymbol{p}, \boldsymbol{q}) - \Pi_{[l,u]^{m\times n}} \left(\boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}\left(\boldsymbol{p}, \boldsymbol{q}\right) \right) \right|\right|_F \right\} \tag{6.37}$$

where all operators and variables are given in definition 6.1.

The next step in solving this dual problem is finding the gradient ∇h, which is given by the following lemma

Lemma 6.3 (Gradient needed to solve TV dual problem [2])
The dual problem consists of minimising the function $h(\boldsymbol{p}, \boldsymbol{q})$ *which has the gradient*

$$\nabla h(\boldsymbol{p}, \boldsymbol{q}) = \lambda \mathcal{L}^T \Pi_{[l,u]^{m\times n}}(\boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}(\boldsymbol{p}, \boldsymbol{q})) \tag{6.38}$$

where all operators and variables are given in definition 6.1.

Solving the dual problem requires us to solve a minimisation problem with linear constrains, the way we do that is by using the orthogonal projection operator $\Pi_{\mathcal{P}}$ at every iteration to make sure we are looking for the correct solutions. We want to solve this problem with FISTA without backtracking, and to do that we need a maximum bound for the Lipschitz constant $L_{\nabla h}$, which is given by

$$L_{\nabla h} \leq 4\lambda^2. \tag{6.39}$$

This equation gives us algorithm 9 for computing the TV denoising problem. We can see, from Beck and Teboulle's paper, that this converges fast, and that about 30 iterations are more than enough. [2] We won't use a restart scheme here, since it converges both quickly and well without it.

Algorithm 9 FISTA for TV denoising [3]

Initialise:
$\boldsymbol{p}_1 = \boldsymbol{r}_1 = 0_{(m-1)\times n}$
$\boldsymbol{q}_1 = \boldsymbol{s}_1 = 0_{m\times(n-1)}$.
$t_1 = 1$
for $k \leftarrow 1, N$ **do**
$(\boldsymbol{p}_k, \boldsymbol{q}_k) \leftarrow \Pi_{\mathcal{P}}\left[(\boldsymbol{r}_k, \boldsymbol{s}_k) + \frac{1}{4\lambda}\mathcal{L}^T\left(\Pi_{[l,u]^{m\times n}}\left[\boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}(\boldsymbol{r}_k, \boldsymbol{s}_k)\right]\right)\right]$
$t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k}}{2}$
$(\boldsymbol{r}_{k+1}, \boldsymbol{s}_{k+1}) = (\boldsymbol{p}_k, q_k) + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{p}_k - \boldsymbol{p}_{k-1}, \boldsymbol{q}_k, \boldsymbol{q}_{k-1})$
end for
$\boldsymbol{x}_n \leftarrow \Pi_{[l,u]^{m,n}}(\boldsymbol{x}_0 - \frac{\lambda}{2}\mathcal{L}(\boldsymbol{p}_k, \boldsymbol{q}_k))$
return $\boldsymbol{x}_n$

The next thing to do is to find the proximal gradient of the TV regularised inverse problem, Beck and Teboulle propose, without proof, that:

Proposition 6.3
The modified proximal map of the TV-regularisation technique $P_{TV}(\boldsymbol{x})$ is given by

$$P_{TV}(\boldsymbol{x}; L) = Dc\left(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x}), \frac{2\lambda}{L}\right) \tag{6.40}$$

where Dc is the TV denoising problem and $r(\boldsymbol{x})$ is the residual of our image reconstruction problem.

Proof.

$$Dc(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x}), \frac{2\lambda}{L}) = \arg\min_{\boldsymbol{y}} \left\{ \left|\left| \boldsymbol{y} - \left(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x})\right)\right|\right|_2^2 + \frac{2\lambda}{L}||\boldsymbol{x}||_{TV} \right\} \tag{6.41}$$

We can multiply everything within the $\{\}$ by $\frac{L}{2}$ since the minimizer don't change when the function is multiplied with a constant.

$$Dc(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x}), \frac{2\lambda}{L}) = \arg\min_{\boldsymbol{y}} \left\{ \frac{L}{2}\left|\left| \boldsymbol{y} - \left(\boldsymbol{x} - \frac{1}{L}\nabla r(\boldsymbol{x})\right)\right|\right|_2^2 + \lambda||\boldsymbol{x}||_{TV} \right\}. \tag{6.42}$$

This is equal to the proximal gradient. □

This means that we have to solve a optimisation problem every iteration. The runtime of this optimisation is, surprisingly enough, not noticeably longer than for L_1 and Tikhonov regularisation. The results from this technique are, as we can see in figure 6.10 and 6.11, really good, with large regions without variation in attenuation. It is however important to note that TV regularisation creates, with very noisy images, larger artifacts than filtered backprojection. There are fewer artefacts, but they are larger in actual size, something that might not be ideal for certain applications. The other main problem with this kind of regularisation is time and that very small details might be removed in order to induce gradient sparsity.

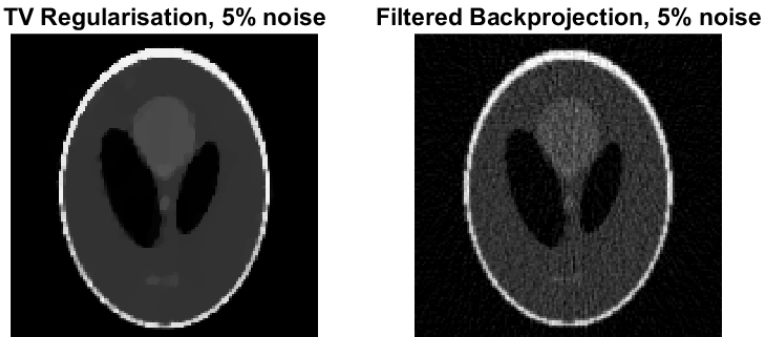

Figure 6.10: Both images are reconstructions with 5% noise. To the left: TV regularisation with $\lambda = 0.15$. To the right: Filtered Back Projection with Shepp-Logan filter

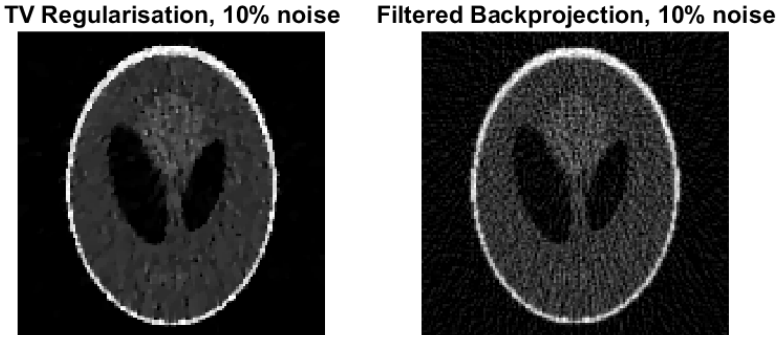

Figure 6.11: Both images are reconstructions with 10% noise. To the left: TV regularisation with $\lambda = 0.175$. To the right: Filtered Back Projection with Shepp-Logan filter

7 Discussion of numerical experiments

The first experiment regarding our noise model showed a clear difference between normally distributed noise and the noise model we used. This is an important observation to make for the testing and development of new reconstruction methods, as developing an algorithm that only works with white noise would not be desirable. It would obviously be best to use real data for such purposes, but real data is often difficult to get hold of, so high quality noise simulations should therefore be conducted.

It is, however, important to consider that this noise model assumes mono-energetic photons, whereas a true X-ray source will generate a spectrum of energies, with different peaks. It is therefore worth considering the generation of a noise model that considers multiple energies with corresponding attenuation, as attenuation is dependent on photon energy.

Nevertheless, spending too much time developing a perfect noise model might not be ideal, as it is not certain whether or not the effects such a model has on reconstructed images are detrimental, and may thus detract from generating new reconstruction methods. I therefore propose testing regularisation techniques with both white noise and our noise model (maybe with different intensities), instead of spending a long time creating a complicated, and more accurate, model.

The numerical experiments of regularisation techniques had clear results, and there were two methods that stood out; filtered backprojection for speed and TV regularisation for noise reduction. Filtered backprojection took seconds to compute, and there is, therefore, no good reason *not* to reconstruct an image this way, if not in addition to using other reconstruction algorithms.

We used the Shepp-Logan phantom as seen in 1.1, and this phantom has large constant areas with few small details. Our simulations might therefore not be ideal for all CT scans. This is an effect we expect to see the most in TV regularisation, as that method works best for such images, it is therefore worth conducting more experiments with a wide range of phantoms in order to better compare TV regularisation and filtered backprojection.

L_1 and Tikhonov regularisation did poorly in our experiment, but this is not a surprise as they are general regularisation techniques. They were in other words not chosen for any specific property other than generating a well posed problem. L_1 regularisation gave us a sparse image, which is both relieving and expected, and it encourages looking for a basis in which CT-images are sparse.

Tikhonov regularisation did, as discussed earlier, not have much of a positive effect. We know that Tikhonov regularisation penalises high pixel values much more than low pixel values since it penalises the 2-norm of our image. As a result of this, Tikhonov regularisation will reduce the intensity of bright pixels without creating too big of a residual, which again will increase the value of

dark pixels. It will, therefore, smooth the picture.

Using Tikhonov regularisation after that explanation might sound like a good idea, but it will also blur the image in the process of naively smoothing it. It seems better to use a method that smoothes the image whilst retains sharp edges, which is, as mentioned earlier, what TV-regularisation does. By inducing a sparse gradient we get a picture with large monotone regions and sharp edges and that is exactly what we want.

Some might suggest using the energy norm, $||\nabla x||_2$, instead of the TV norm as regularisation term. This is not ideal as minimising the 2-norm of the gradient will smooth the gradient, the same way Tikhonov regularisation smoothes images, and remove peaks in the gradient. This should in turn generate a blurry image with slow changes in pixel intensities. It will also, in large part, ignore noise because noise corresponds to small changes in pixel intensities. The gradient will therefore have a small 2-norm and thus, in large part, be ignored by the regularisation term. The solution to this problem is getting a sparse gradient, which is attained by TV regularisation.

Another thing to consider further is combining the images gained by TV regularisation with images gained by filtered backprojection, as some details might be easier to see in using TV regularisation than filtered backprojection and vice versa. Small details are especially prone to disappear with TV regularisation. Using a combination of both techniques, either side by side or overlayed images might therefore be ideal in some situation.

The main problem with TV-regularisation is, as mentioned earlier, the time it takes to compute. There are however multiple different ways of solving this. Our first observation is that the discrete Radon transform and/or the backprojection are the most computationally heavy parts in our algorithms. The way we notice this is that it takes about the same time to do 100 iterations as Tikhonov regularisation as TV regularisation, which we know requires solving an optimisation problem for each iteration. This means that the runtime of our iterative algorithms would decrease substantially if we were able to find a more efficient way of computing the discrete Radon transform and discrete backprojection operator.

7.1 Further work

This is a field where there still is both research and work left, the most notable is getting an efficient Radon transform and backprojection algorithm. These transformations can be written as matrix transformations, and it is well known how to solve such problems in parallel on a graphics card. This might be the first step towards getting TV regularisation for public use. Another thing to consider is using denoising algorithms on the sinogram before reconstructing it with TV-regularisation.

Bibliography

[1] H.H. Bauschke and P.L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces. CMS Books in Mathematics. Springer New York, 2011.

[2] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Transactions on Image Processing, 18(11):2419–2434, November 2009.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal of Imaging Sciences, 2(1):183–202, 2009.

[4] Jack Dongarra and Francis Sullivan. Guest editors' introduction: The top 10 algorithms. Computing in Science & Engineering, 2(1):22–23, 2000.

[5] Heinz W. Engl, Martin Hanke, and Andreas Neubauer. Regularization of Inverse Problems. Kluwer Academic Publishers, 2000.

[6] John Erdos. Lecture notes in "operators on hilbert space". Accessed 27 April 2016].

[7] Aaron G Filler. The history, development and impact of computed imaging in neurological diagnosis and neurosurgery: Ct, mri, and dti. Nature Precedings, 7(1):1–69, 2009.

[8] D.C. Lay. Lay:Linear Algebra and Its Applications: International Edition/ Student Study Guide for Linear Algebra and Its Applications. Pearson Education, Limited, 2012.

[9] A. Macovski. Medical Imaging Systems. Prentice-Hall information and system sciences series. Prentice-Hall, 1983.

[10] Frank Natterer. The Mathematics of Computerized Tomography. John Wiley & Sons Ltd., 1986.

[11] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In Doklady an SSSR, volume 269, pages 543–547, 1983.

[12] Brendan O'Donoghue and Emmanuel Candès. Adaptive restart for accelerated gradient schemes. Foundations of Computational Mathematics, pages 715–732, 2013.

[13] Neal Parikh and Stephen Boyd. Proximal algorithms. Foundations and Trends in Optimization, 1(3):127–239, 2014.

[14] Geoffrey J.M. Parker. Lecture notes in "physics of medical imaging", Spring 2016.

[15] Johann Radon. On the determination of functions from their integral values along certain manifolds. Medical Imaging, IEEE Transactions on, 5(4):170–176, 1986.

[16] Carlos Ramirez, Vladik Kreinovich, and Miguel Argaez. Why l1 is a good approximation to l0: A geometric explanation. Journal of Uncertain Systems, 7(3):203–207, 2013.

[17] Steven Roman. Advanced Linear Algebra. Graduate Texts in Mathematics. Springer, 3 edition, 2007.

[18] J.R. Taylor. An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements. University Science Books, 1997.