Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Ludvigsen, Martin

Parameter selection for total variation regularization with applications in imaging

Master's thesis in Applied Physics and Mathematics

Supervisor: Grasmair, Markus

August 2020

NTNU
Norwegian University of Science and Technology

Abstract

Regularization methods are commonly used to solve inverse problems. In particular, total variation regularization has proven to be effective for image denoising. Regularization methods usually require selection of one or more numerical parameters in order to be effective, which can be a challenge in many applications. In this thesis, we give an introduction to Fourier analysis and convex analysis, which is needed to understand and implemnet total variation denoising and deconvolution. Building on recent work, we propose a new method for parameter selection which we call Solution Regularization. The method consists of choosing the reconstruction that is closest to an approximate solution, which is found by another reconstruction method. We compare our method with existing methods and show that our method can give superior results for image denoising and image deconvolution.

Sammendrag

Regulariseringsmetoder er ofte brukt for å løse inverse problemer. Spesielt har total variasjonregulering vist seg å være effektiv for støyfjerning av bilder. Regulariseringsmetoder krever vanligvis valg av én eller flere numeriske parametere for å være effektiv, og dette kan være utfordrende i mange applikasjoner. I denne avhandlingen gir vi en introduksjon til Fourieranalyse og konveks analyse, som trengs for å forstå og implementere total variasjon støyfjerning og dekonvolusjon. Vi foreslår en ny metode for parametervalg som bygger på nyere arbeid som vi kaller Solution Regularization (Løsningsregularisering). Metoden består av å velge rekonstruksjonen som er nærmest en tilnærmet løsning, som vi finner med en annen rekonstruksjonsmetode. Vi sammeligner våre resultater med eksisterende metoder of viser at vår metode kan gi bedre resultater for både støyfjerning og dekovolvering av bilder.

Acknowledgements

This thesis was written during a particularly turbulent time, under difficult circumstances. Yet, there is always room to be thankful.

I would first like to give my warmest thanks to my supervisor Markus Grasmair. In spite of the difficult situation, he kept me motivated and helped me finish this work. Although this thesis consists mostly of my own thoughts, he has taught me most of it and shaped the rest, and I am thankful for all the help, useful comments and lessons I have learned along the way. I look forward to the next few years where he will continue to be my supervisor on my PhD thesis.

Secondly, I would like to thank Valeriya Naumova and Zeljko Kereta from Simula Research Laboratory who helped me in the initial phases of this work. Most of the contributions in this thesis is based on their prior work, so this thesis would not exist had it not been for them.

Thirdly, I would like to thank the Norwegian University of Science and Technology, especially the Department of Mathematical Sciences. Knowledge is perhaps the greatest wealth, and by that measure they have made me rich man. They are also my future employers, and I look forward to the years that lie ahead.

I would lastly like to express my gratitude to my parents. Not only have they given me the gift of life, they have given me all the support in the world during this time that has been difficult for all of us. This is something I will forever be grateful for.

Table of Contents

Chapter 1

Introduction

In science and engineering, solving **inverse problems** is the art of recovering data from noisy, corrupted measurements. In many settings, one uses data and a mathematical model to make predictions. When solving inverse problems, one goes the other way around, which is where the word ”inverse” comes from. The field saw great advances in the the 1960s, and was mainly of interest to geophysicists [2]. Since then, the field of inverse problems has grown relatively quickly, and has found its use in many applications. One application of interest in this text is image analysis, where inverse problems can be used to model image reconstruction.

Traditionally, inversion theory uses techniques from Bayesian statistics and/or functional analysis. In text we will use techniques from the latter, specifically using tools from regularization theory. Here, data is reconstructed by solving a well-chosen optimization problem using some knowledge about the data. In this setting another problem has to be solved, namely the problem of parameter selection. The optimization problem usually has one or several numerical parameters that can be changed to yield different reconstructions. Choosing this numerical parameter has traditionally been done using heuristic methods.

In this master thesis, we will investigate the parameter selection problem, building on ideas from Vito et al. [9]. In their work, they suggest an unsupervised method for parameter selection in elastic net regularization of images. In this text we will try to apply the same method to total variation regularization, specifically to solve the denoising problem and deconvolution problem for images. In order to achieve this, we will take a tour through Fourier analysis and signal processing, image resampling and convex analysis.

Almost all the code used to produce this thesis can be found freely available on GitHub: https://github.com/martilud/Master-Thesis. The code is written in Python and FORTRAN using NumPy. This code will also be used in the initial stages of further work after this thesis, and is therefore subject to change.

The text is built up in the following way: Chapter 2 consists of an introduction to image analysis, inverse problems, regularization and specifically total variation regularization. Chapter 3 consists of an introduction to Fourier analysis and signal processing, with the aim of understanding its role in solving inverse problems and convolutions. Chapter 4

consists of an introduction to image resampling, which we will use to develop methods for the parameter selection problem. Chapter 5 consists of an introduction to convex analysis, which is used to formulate numerical methods for solving total variation regularization problems. Chapter 6 contains a summary of some traditional parameter selection methods, ending with a numerical experiment showcasing the different methods. Chapter 7 is the main contribution of this work, consisting of a short statement of the method proposed by Vito et al. [9], followed by some extensions of the method with numerical experiments. This part mostly exhibits the potential of the proposed methods. Chapter 8 contains a summary of how the proposed methods can be investigated further, which will be one of the pursuits of an upcoming PhD thesis.

Chapter 2

Images and inverse problems

This chapter will give an introduction to digital image processing, introduce inverse problems and explain its role in image processing. Near the end we will introduce Total Variation (TV) regularization, which will be the main topic of discussion in this text.

2.1 Digital image processing

Digital image processing is the use of a digital computer to process digital images. It is a subfield of digital signal processing [20]. Digital image processing has been a fruitful field since the advent of the digital computer. It has benefited from the development of many mathematical fields: Fourier analysis, functional analysis, high-dimensional probability theory and machine learning, to name a few. The demand for sophisticated and robust image processing algorithms is prevalent in many scientific and industrial fields.

A rectangular, two-dimensional, grayscale, discrete digital **image** is a signal u with components u_{ij} for $i = 1, ..., n_x, j = 1, ..., n_y$. In other words $u \in \mathbb{R}^{n_x \times n_y} \sim \mathbb{R}^{n_x n_y}$.

The components are usually called **pixels**, whose values describe the intensity of light in this part of the image. Pixels can also be thought of as samples of the original light they attempt to capture.

The pair of integers n_x and n_y describes the size of the image, and is called the **resolution** of the image. We will denote the resolution of an image as $n_x \times n_y$. In this text, we will focus on square, grayscale images to make our life easier. From now on we denote the size of images with $n = n_x = n_y$.

When storing images, one also has to consider the **bit depth** of the image, which describes how many bits are used to store each pixel of the image. Grayscale images are usually stored in 8-bit, meaning that each pixel is allowed to take on 256 different values for intensity. In this text, we will always rescale pixels to be floating point numbers between 0 and 1.

In studying images, it is often useful to look at images as functions $u : \Omega \to \mathbb{R}$ usually belonging to a suitable Banach or Hilbert space, where $\Omega \subset \mathbb{R}^2$. In this text we will select $\Omega = [0, n_x - 1] \times [0, n_y - 1]$. This allows us to use powerful techniques from integral

and differential calculus. In this sense a discrete image contains pixels sampled from a continuous signal. The standard convention for image coordinates is to start at the top left corner, which one has to be careful with in implementations. An illustration of this is shown in figure 4.1.

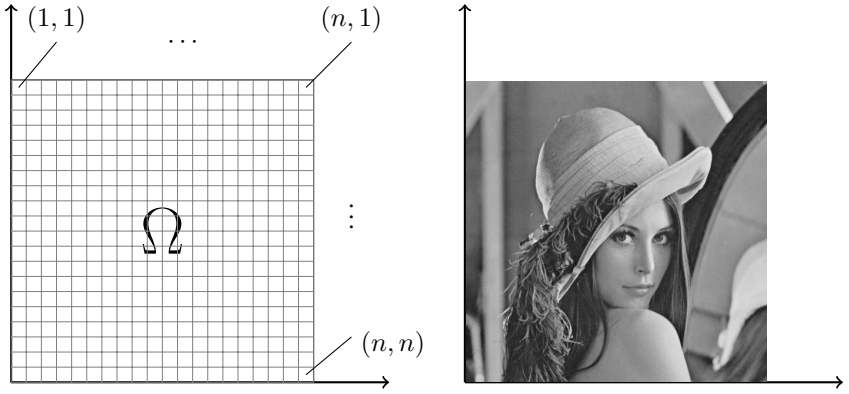

Figure 2.1: To the left we have a domain $\Omega \subset \mathbb{R}^2$ with image coordinates on a grid. To the right is a grayscale image $u : \Omega \to \mathbb{R}$.

In this text we will mainly discuss natural images, which in this text are defined as images of natural scenes, taken with a camera. The method of capture naturally changes some of the properties of the resulting image. Some other classes of images that has seen many applications of image processing are the fields of medical imaging and astronomic imaging. It is clear that the space of natural images has some structure, but it is very hard to mathematically describe this structure. There are certainly many "images" in $\mathbb{R}^{n\times n}$ that many will not consider natural. We will approach images mostly as an element of a vector space, or a discrete signal, not making any more assumptions on the structure of the images.

2.1.1 Inverse problems

Inverse problems are concerned with the recovery of an unknown quantity from a corrupt observation. The process of corruption can be complicated, usually consisting of non-linear physical phenomenon and intricate noise. The relationship between the observation and the unknown quantity that we will use is

$$v = Au + \sigma w, \tag{2.1}$$

where $v \in V$ is the observed data, $u \in U$ is the original data, $w \in V$ has components w_i independent and identically distributed with $\mathrm{E}(w_i) = 0$ and $\mathrm{Var}(w_i) = 1$, meant to model noise, with noise level $\sigma > 0$ and $A : U \to V$ is a linear forward operator. The spaces V and U are usually Banach or Hilbert spaces. The inverse problem thus consists of solving a linear equation with some added noise. The linear operator A describes the forward model that our data follows. Choosing A is usually done based on some information about

the data, and in this text we will only consider cases where A is known. Having some uncertainty in the forward model A can make solving the inverse problem significantly harder. We will often use some convenient abuse of notation when describing the noise vector w, as it represents both a random variable and an actual realization.

General inverse problems need not follow equation 2.1. In particular, the forward operator A can be non-linear (which it almost certainly is if it models some physical phenomenon). In this case, a linear approximation of the forward operator can be used to simplify the problem. Furthermore, additive, i.i.d noise is rarely realistic, but is a useful model.

Many problems in image processing can be formulated as inverse problems depending on the linear operator A. Examples are the denoising problem, where $A = \text{Id}$, deconvolution when A is a convolution operator and inpainting when A is a masking operator.

Noise is an unwanted modification of a signal. It is often considered random, and can also be thought of as the part of the signal carrying no useful information. Noise occurs in signal processing during capture, storage and transmission etc. of signals. Noise is so prevalent, that one can almost always assume that whenever there is a signal, there will also be some unwanted noise. There are many different types of noise that can be relevant in image processing. Noise may arise from random behaviour of particles, down to the statistical mechanical and quantum mechanical level, the image capturing technology itself, errors in analog to digital conversion and compression/decompression to name some mechanisms [4]. The different kinds of noise are usually named after the statistical model they follow. The most relevant type of noise in all of signal processing is **Gaussian noise** where the probability density function (PDF) of each element has a Gaussian PDF

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp -\frac{x^2}{2\sigma^2}. \tag{2.2}$$

Here μ is the expected value which coincides with the location parameter and σ is the standard deviation which coincides with the scale parameter.

A type of noise that is inevitable in many systems is so-called **shot noise**. This type of noise occurs in images because of the particle behaviour of electromagnetic waves, and in electrical circuits because of the particle nature of electrons. This makes this kind of noise inevitable in images and especially in medical images. When measuring the intensity of a pixel one is essentially measuring the flux of electromagnetic particles, i.e how many particles pass a surface per time. The actual amount of particles can vary because of the statistical nature of particles. The amount of particles measured is usually modeled by a Poisson process, which means the noise follows a Poisson distribution. When the amount of particles is very large the difference from the mean becomes relatively small. Thus, when the amount of particles is low, the shot noise is more prevalent, but as the amount of particles becomes larger the noise becomes more Gaussian and less prevalent, as the Poisson distribution approaches the Gaussian distribution. This is an example of why Gaussian noise is very common. Simulating this kind of noise can get technical, and will look almost identical to Gaussian noise for realistic examples.

There exists more ”exotic” noise models, like noise resulting from more complex stochastic processes and structurally dependent noise. In this work we will only consider additive Gaussian noise. When adding Gaussian noise to an image we will project the

resulting noisy image so that the actual pixel values lie between the minimal and maximal allowed pixel value. We do this because it is more realistic and does not alter the contrast of the noisy image. This makes it easier to compare noiseless and noisy images. It also makes our noise model slightly incorrect. We will ignore this inaccuracy.

Some examples of Gaussian noise is shown in figure 3.9. We will refer back to these examples several times in this text.

Figure 2.2: Original image and image with added Gaussian noise with $\sigma = 0.05, 0.075, 0.1$ respectively.

Convolutions are, in the context of image processing, an operation between an image and a kernel or mask κ. They are used for many tasks, and are often synonymous with applying a filter to the image. They are a subclass of more general linear operators. We will describe convolutions more thouroughly in a later section. Some common filters are Gaussian blurring (which has applications in smoothing and removing noise), edge detection and sharpening filters. Physical phenomena can also cause blur. Lenses are seldom perfect or perfectly focused, which leads to blur in our human vision as well as cameras. Cameras have a shutter speed, meaning the time where light reaches the sensor of the camera to cre-

ate a photo , and this time is obviously non-zero. If the light comes from a moving source relatively to the camera during this time, the image can be corrupted with what is called motion blur. When light travels through the atmosphere, refraction can cause blur, usually called atmospheric blur. These effects are very relevant in all imaging, from medical to atrological, which is why we need methods for removing blur.

Some examples of different linear operators leading to different image corruptions is shown in figure 2.3.

Figure 2.3: Different image corruptions corresponding to different forward operators A. The corruptions are made using a Gaussian blur filter, a masking operator and a laplace filter respectively.

When talking about noise and corruptions it is useful to have some quantity describing how noisy and corrupted an image is. Conversely, we need a quantity to describe how similar two images are. A simple choice for this is the Mean Squared Error (MSE). Given an original image u and noisy image v with resolution $n \times n$, we define the MSE as

$$\mathrm{MSE}(u, v) = \frac{1}{n^2} \sum_{i,j} (u_{i,j} - v_{i,j})^2 = \frac{1}{n^2} \|u - v\|^2. \tag{2.3}$$

A problem with the MSE is that it is dependent on the intensity range of the images. A different quantity is the Peak signal-to-noise ratio (PSNR), which can be defined on a decibel scale as

$$\mathrm{PSNR}(u, v) = 10 \log_{10} \left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}(u, v)} \right), \tag{2.4}$$

where MAX is the maximal value attained by a pixel of the original image. For 8-bit images, we have $\mathrm{MAX} = 255$. Higher PSNR means the original and noisy images are more similar. If nothing else is specified, the value of the PSNR is usually between a noisy or reconstructed image and the orignal image.

There exists other signal-to-noise variants. One problem with using PSNR is that it measures distance between two images differently from how humans measure similarity. PSNR only measures the pixelwise intensity difference. For example, a shifted or rotated image can yield low PSNR even though the images carry much of the same information. Another example is smaller structures in an image, like objects or parts of an object.

A quantity that can be used to measure the structure of an image is the so-called structural similarity index measure(SSIM) [25]. The SSIM compares three measures of a smaller subset of the images, usually called a window of the images. The three measures

are intensity(I), contrast (C) and structure (S). Given two windows x and y of equal size, the three quantites are measured by

$$I(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1},$$
$$C(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2},$$
$$S(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3},$$

where μ_x and σ_x is the mean and standard deviation over the window and σ_{xy} is the covariance between the two windows. The constants c_1, c_2 and c_3 are there to stabilize the division, in case denominators are close to 0. These three constants are usually chosen as $c_1 = (k_1 \operatorname{MAX})^2$ with $k = 0.01$, $c_2 = (k_2 \operatorname{MAX})^2$ with $k = 0.03$ and $c_3 = 0.5(k_2 \operatorname{MAX})^2$. The SSIM can be attained by multiplying the three quantities to obtain

$$\operatorname{SSIM}(x,y) = \frac{(2_x\mu_y + (k_1 \operatorname{MAX})^2)(2\sigma_{xy} + (k_2 \operatorname{MAX})^2)}{\mu_x^2 + \mu_y^2 + (k_1 \operatorname{MAX})^2)(\sigma_x^2 + \sigma_y^2 + (k_2 \operatorname{MAX})^2}. \tag{2.5}$$

The SSIM is a number between -1 and 1, or 0 and 1 if the components of the two windows have the same sign, where the value 1 corresponds to two equal images. The SSIM is non-negativity, but is not a metric as it does not satisfy the triangle inequality. The SSIM for an entire image is usually calculated using a sliding block window, where we choose the size 9×9.

2.1.2 Solving inverse problems

Recovering data u from an observation v in equation (2.1) is an inverse problem. We will only consider finite-dimensional inverse problems. Some concepts and results are also relevant in infinite dimensions, and notation is usually ambiguous.

When there is no noise the inverse problem corresponds to solving the linear equation problem $Au = v$, which on its own can be difficult. From linear algebra we know that a solution exists if $v \in \operatorname{range}(A)$. This is not always the case, and even when $v \in \operatorname{range} A$ there can be infinitely many solutions. A way to circumvent this is to solve a altered problem which has better properties.

A common assumption is that the original data u lies in some well-chosen subspace W. On finite-dimensional Hilbert spaces this can be done by solving the least squares problem

$$Sv = \operatorname*{arg\,min}_{u \in W} \|Au - v\|_2^2. \tag{2.6}$$

Here we define the **solution operator** S, which maps the observation to a proposed solution. When noise is added, the inverse problems coresponds to solving $Au = v - \sigma w$, but we generally do not know what σw is. Thus, trying to obtain a reconstruction in spite of the noise is the goal.

There are three main difficulties with inverse problems, and the point with an inversion method like the least square solution should be to alleviate some of these problems.

Firstly, **existence** of a solution. A solution only exists if $v - \sigma w \in \text{range}(A)$. A least squares solution can be shown to exist. Secondly, **uniqueness** of a solution. If the kernel of the operator A is non-trivial, we can always add an element in $\ker(A)$ to a solution to achieve another solution, meaning there can be infinitely many solutions. A least squares solution is not necessarily unique. Lastly, the **stability** of a solution. The stability of a linear operator A is often measured with the **condition number** of the operator. If we let δ be a perturbation (like noise) in the data v, then the condition number is defined as

$$\mathcal{K}(A) = \max_{v,\delta \neq 0} \frac{\|A^{-1}\delta\|}{\|\delta\|} \frac{\|v\|}{\|A^{-1}v\|} = \|A^{-1}\|\|A\| \geq 1. \tag{2.7}$$

In the case where A^{-1} does not exist, one usually interprets this as the condition number being infinite. We can use the condition number as an upper bound of how much perturbations in the data affect the solution of $u + \varepsilon = A(v + \delta)$. Here ε is the error that occurs when perturbing the data v. Specifically, one finds that

$$\frac{\|\varepsilon\|}{\|u\|} \leq \mathcal{K}(A) \frac{\|\delta\|}{\|v\|}. \tag{2.8}$$

Many inverse problems are ill-conditioned, meaning that the condition number is high, or infinite for non-invertible A. Some linear problems are so ill-conditioned that they cannot be solved without taking precautions because of machine precision. When noise or other perturbations are added, the instability of the inversion can yield wildly innacurate solutions.

2.2 Regularization

Inversion methods can be significantly improved by adding regularization. The main idea of regularization is to constrict the solution to lie on a subset with suitable properties, which usually means solving an altered problem. For our least squares problem, we can choose a **regularization functional** $R : U \to \mathbb{R}^+$ that chooses some property we want the inversion to have, and define

$$Sv = \arg\min_u \frac{1}{2}\|Au - v\|_2^2, \quad \text{subject to } R(u) \leq c, \tag{2.9}$$

where c is a non-negative scalar. Here the term $\frac{1}{2}\|Au - v\|_2^2$ is usually called the **data discrepancy** term, which measures the distance between the observed and reconstructed data. We can also write this regularization problem on the lagrangian form

$$S(\lambda)v = \arg\min_u \underbrace{L(u, v)}_{\text{Data discrepancy}} + \underbrace{\lambda R(u)}_{\text{Regularization}}. \tag{2.10}$$

where the Lagrange multiplier $\lambda \geq 0$ is called the **regularization parameter**. We use the notation $S(\lambda)v$ to denote a solution operator depending on a parameter λ, acting on v.

The data discrepancy term is usually chosen based on information about the noise. The commonly used data discrepancy $\|Au - v\|_2^2$ can be derived as the log-likelihood of u given Gaussian noise.

We will now Let D be a linear operator that selects some property we want to regularize. Some commonly used regularizers are:

- L_1-Regularization $R(u) = \|Du\|_1$.
- L_2-Regularization $R(u) = \|Du\|_2^2$.
- Elastic net regularization $R(u) = \|Du\|_1 + \alpha\|u\|_2^2$, where $\alpha > 0$ is an additional parameter.

For example, if we let $D = I$, L_1-regularization is usually called Lasso regression and L_2-regularization is usually called Ridge regression in the statistical learning framework [**?**]. Both Lasso and Ridge add bias, which by the bias-variance trade-off should reduce the variance and reduce overfitting. Lasso also does variable selection, meaning it completely eliminates some elements of the solution. This is an attractive property in many research applications. Elastic net is a mix of Lasso and Ridge regression [**?**], and also does variable selection.

2.2.1 Why regularization?

We will attempt to give a short overview of why regularization can yield good solutions to inverse problems, starting with a common L_2 example. We define the solution operator

$$S(\lambda)v = \frac{1}{2}\|Au - v\|_2^2 + \frac{\lambda}{2}\|Du\|_2^2. \tag{2.11}$$

We can apply the Euler-Lagrange equation to find that the solution operator coincides with the solutions to

$$A^*(Au - v) + D^*Du = 0. \tag{2.12}$$

The solution operator can then be written as

$$S(\lambda)v = (A^*A + \lambda D^*D)^{-1}A^*v, \tag{2.13}$$

when $(A^*A + D^*D)$ is invertible. Note that the solution operator is linear in v and continuous and smooth in λ.

The operators A^*A and D^*D are always semi-positive definite, which is easy to see

$$u^*A^*Au = (Au)^*Au = \|Au\|_2^2 \geq 0. \tag{2.14}$$

While this does not guarantee that A^*A or D^*D is invertible, $A^*A + \lambda D^*D$ will be invertible if either A or D is invertible. Thus, if the least square solution $u = (A^*A)^{-1}A^*v$ does not exist, the L_2 regularized solution will exist if D^*D is invertible. Conversely, if the regularizer is not well-posed, we require that A^*A is invertible for a unique solution to exist.

If we assume that A^*A and D^*D have the same eigenspace, which we will see is a valid assumption in this work, then $A^*A + \lambda D^*D$ will have eigenvalues $\lambda_i^A + \lambda\lambda_i^D, i = 1, ...n^2$. It can be shown that the condition number satisfies [22]

$$\mathcal{K}(A^*A + \lambda D^*D) = \frac{\max_i |\lambda_i^A + \lambda\lambda_i^D|}{\min_i |\lambda_i^A + \lambda D_i|}. \tag{2.15}$$

Thus, if A^*A is ill-conditioned, meaning the ratio between the largest absolute eigenvalue and the smallest absolute value is large, it can become better conditioned by adding λD^*D if λD^*D has a smaller spectrum. If this is the case, the problem becomes better conditioned as λ increases, which corresponds to the problem becoming more regularized. However, a better-conditioned problem does not necessarily corresponds to a better solution to an inverse problem. As the problem becomes more regularized, we stray away from the original problem $Au = v$, and the regularized solution becomes biased. We can also interpret the regularization procedure as a preconditioner of the problem $Au = v$, where we have

$$S(\lambda)v = S(\lambda)Au = (A^*A + \lambda D^*D)A^*Au. \tag{2.16}$$

If we add noise to the problem, so that $v = Au + \sigma w$, we have by the linearity of the solution operator

$$S(\lambda)v = S(\lambda)Au + \sigma S(\lambda)w. \tag{2.17}$$

In this case, we do not only want to stabilize the problem $Au = v$, but we also want to choose our regularizer so that the term $\sigma S(\lambda)w$ vanishes. The act of removing noise can be difficult because there is a trade-off between removing noise and keeping the signal unaltered.

It is also clear that for larger σ, i.e more noise, we need more regularization for the noisy term to vanish. The main difficulty of choosing a regularizer, and the regularization parameter λ is to balance the removal of noise and the stabilization of the problem $Au = v$ without adding too much bias and corruption to the reconstruction u. We note that for a linear solution operator, and if we interpret w as a random vector with independent elements, zero mean and variance equal to unity, we have $\mathrm{E}(\sigma S(\lambda)w) = 0$ for any linear operator $S(\lambda)$. As for the variance, we have $\mathrm{Var}(S(\lambda)w) = \sigma^2 S(\lambda)^* S(\lambda)$. We immediately note that unless the operator $S(\lambda)$

For the L_1 problem, we have the solution operator

$$S(\lambda)v = \frac{1}{2}\|Au - v\|_2^2 + \lambda\|Du\|_1. \tag{2.18}$$

Applying the Euler-Lagrange equation we find that this corresponds to

$$A^*(Au - v) + \lambda D^* \frac{Du}{\|Du\|_1} = 0. \tag{2.19}$$

This equation is non-linear, and the solution operator is non-linear and we cannot find a closed form operator in the general case. This makes everything more difficult. However, in many cases, this solution operator has many similar properties to the L_2 case.

In the regularization setting, the inversion problem has two parts. The choice of a suitable solution operator and the choice of the parameters of the solution operator. We focus mainly on the unsupervised case, i.e when the original data u is not known. In this work, we consider the regularization method as given, and focus our energy on choosing the correct regularization parameter. This means that many of the methods proposed can be used on many different problems . In imaging, one of the regularization functionals that have seen the most success is **total variation** regularization, which will be the main focus of this work. We will also focus on a cousin of total variation, namely **quadratic variation**.

2.3 Quadratic Variation and Total Variation regularization

Before we discuss total variation we will discuss a similar problem which will be helpful in this work, quadratic variation regularization. Quadratic variation regularization corresponds to L_2 regularization of the gradient of the reconstruction

$$S_{\text{QV}}(\lambda)v = \min_{u \in U} \frac{1}{2}\|Au - v\|_2^2 + \frac{\lambda}{2}\||Du|\|_2^2, \tag{2.20}$$

where D is a gradient operator. In the continuous case D is the gradient operator $\nabla : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$. In the discrete case, a common gradient operator is $D : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2 \times 2}$

$$(Du)_{i,j} = \left(\begin{cases} u_{i+1,j} - u_{i,j}, & i < n \\ 0, & i = n \end{cases}, \begin{cases} u_{i,j+1} - u_{i,j}, & j < n \\ 0, & j = n \end{cases} \right), \tag{2.21}$$

which corresponds to forward differences with Neumann conditions on two sides of Ω.

We have that D^*D is the negative (discrete) Laplacian operator with specific boundary conditions, which means that a solution to the quadratic variation problem is equivalent to the solution of the so-called inhomogeneous Helmholtz equation. For slightly differently defined D, the operator D^*D becomes a convolution usually called a Laplacian filter. The aim of quadratic variation regularization is to punish large variations in an image. Large variations are sometimes parts of images, especially edges, but can also be caused by noise. We will later see that the act of punishing large variations is very similar to the idea of a low-pass filter.

It is clear that the solution operator $S_{\text{QV}}(\lambda)v$ is linear in v, as well as continuous and smooth in λ. Finding $S_{\text{QV}}(\lambda)$ can be done computationally efficiently. In this work we will use the conjugate gradient method, but there exists a wealth of effective methods. In particular, if we interpret D^*D as a convolution with a Laplacian kernel, which means that $S_{\text{QV}}(\lambda)$ can be calculated as a convolution, which we will get into later.

Examples of QV denoised images are shown in figure 2.4. We see that using the QV solution operator we are able to significantly remove noise, at the cost of blurring the image. For higher λ, it is very evident that this blurring especially damages the edges of the image. This is the case for most linear denoising operators.

Total variation can be defined in several ways. For smooth functions of two variables it is defined as

$$\text{TV}(u) = \int_\Omega \|\nabla u\| = \int_\Omega \sqrt{\frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2}. \tag{2.22}$$

For 2D discrete signals one version of total variation is

$$\text{TV}(u) = \sum_{i<n, j<n} \sqrt{|u_{i+1,j} - u_{i,j}|^2 + |u_{i,j+1} - u_{i,j}|^2} = \||Du|\|_1. \tag{2.23}$$

We choose to use the notation $\||Du|\|_1$ here, by which we mean that the euclidean norm is applied component-wise for each $(Du)_{i,j}$.

Figure 2.4: Original image u, noisy image $v = u + \sigma w$ with Gaussian noise and $\sigma = 0.1$ and QV reconstructions using $\lambda = 1.0, 5.0, 10.0, 50.0$ respectively.

The applications of total variation denoising was first proposed by L.Rudin, S. Osher and E. Fatemi in 1992 [**?**]. They proposed the Rudin-Osher-Fatemi model for denoising

$$S_{\text{TV}}(\lambda)v = \arg\min_u \frac{1}{2}\|u - v\|^2 + \lambda TV(u). \tag{2.24}$$

In the more general inverse problem case, one would use $\frac{1}{2}\|Au - v\|^2$ as the data discrepancy term.

Lemma 1. *The TV functional on $L_2(\omega)$ is lower semi-continuous and convex.*

Proofs are shown in [5], which is also a much more thorough introduction to total variation and its uses.

While the solution operator is continuous, it is not differentiable with respect to λ everywhere. In fact, it can be shown that for a certain parameter $\bar{\lambda}$, any $\lambda > \bar{\lambda}$ yields a constant solution.

Total variation has proven to be very effective for denoising, but also has uses in some of the other inverse problems mentioned earlier, like deblurring and zooming [6].

Figure 2.5 shows an example denoised images for different regularization parameters. From the figure we see that we are able to remove much of the noise, while also applying a kind of blur to the reconstructions. However, as opposed to the QV reconstructions, the TV reconstructions conserve edges.

For higher regularization parameters, we see that that the reconstruction loses much of the detail by over-smoothing the image, and we see there are ”plateaus” of relatively

Figure 2.5: Original image u, noisy image $v = u + \sigma w$ with Guassian noise and $\sigma = 0.1$ and TV reconstructions using $\lambda = 0.01, 0.05, 0.1, 0.5$ respectively. Note that parameters differ greatly in size from figure 2.4.

similar intensity.

L_1 regularization has a tendency to conserve some componetns of Du while setting others to 0. In the case of TV we see that it conserves large variations, like edges, but smoothes areas with small variations effectively. L_2 regularization does not have this "selective" property to the same degree. TV regularization also yields better values than QV regularization in many cases for images. However, this does not mean that TV is "better" than QV in all cases. In some cases the blur caused by TV can be unwanted as apposed to the blur caused by QV.

From this we can also conclude that choosing the correct regularization parameter is important in order to create a good reconstruction. Furthermore, the optimal parameter (in the sense that it minimizes PSNR) depends on the input data v. We will later see that selecting a good parameter, especially for the TV problem is far from a trivial task. This will be the main issue discussed later in this text.

Solving the TV problem is It is clearly a problem suitable for optimization, but its non-differentiability creates problems for traditional optimization algorithms. A naive guess is to use the Euler-Lagrange equations to obtain the Rudin-Osher-Fatemi PDE:

$$\lambda \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + u - v = 0, \quad u \in \Omega \tag{2.25}$$

$$\frac{\partial u}{\partial n} = 0, \quad u \in \partial\Omega. \tag{2.26}$$

The PDE is highly non-linear and not easy to solve, but this was the approach in the original paper.

Newer methods for solving the total variation problem uses methods from convex analysis, which is what we will use. This will be the topic of chapter 5.

2.4 Image reconstruction in the supervised case

In the supervised case, the goal is to find an optimal solution operator S given a dataset of corrupt observations $v^{(i)}$ and original data $u^{(i)}$. Inverse problems in practice mainly concern the unsupervised case, where only corrupt observation(s) $u^{(i)}$ are given. A different but related categorization is data-driven and knowledge-based methods. Data-driven methods are, as the name suggests, methods where the solution operator is created using observed data. Knowledge-based methods are methods are created using domain-specific knowledge and mathematical modelling. Neural networks, especially convolutional neural networks (CNN) and generative adversarial networks (GAN) are data-driven methods that have been shown to be effective on image denoising and superesolution [16] [29]. In experimental cases, deep learning methods are shown to be more effective than traditional numerical methods. The main strength of deep learning methods is that solution operators can be learnt without much domain knowledge.The main problem of supervised image reconstruction is underfitting and overfitting. In the case of denoising, a good solution operator can be learnt using a few hundred images. Here it is not unreasonable to assume that the solution operator learns an operator that is similar to the methods we will investigate later. For more advanced problems, like superresolution and inpainting, several ten thousand or hundred thousand images are required to create good solution operators. How many images that are required depends on properties of the dataset, like resolution, similarity of images and so on. A recent talk [1] showed that solving the inpainting problem on a dataset of face images can be done using a CNN. A working method requires about eighty thousand image samples in this case. However, if the dataset is reduced to two thousand images, the method completely falls apart, and the CNN just returns noise. In other words, data-driven models can only get you so far, and there is still a need for more knowledge-based models in cases with small amounts of data.

We have so far introduced the main aspects of image reconstruction and regularization. In order to properly tackle the problem of deconvolution, as well as give greater understanding to the methods that will be proposed, the next chapter will give an introduction to Fourier analysis and signal processing.

Chapter 3

Discrete signal processing, Fourier analysis and convolutions

The goal of this section is to give an introduction to Fourier analysis, with the goal of discussing its role in solving inverse problems and discrete signal processing. The main goal is to understand convolutions, image resampling, as well as understanding inversion methods. For images, many corruption processes can be modeled as convolutions, so we will need to understand how we can implement them efficiently. During this endeavor we will also obtain knowledge that will help us understand denoising and the parameter selection problem.

This section is also a look at digital (discrete) signal processing, as viewed by a graduate mathematician. Signal processing concerns **signals**, which is a thing that carries or conveys information [18]. A signal usually exhibits variation in time and/or space, which is why the notion of function or vector from mathematical calculus are commonly used to represent signals.

Signal processing has its roots in mathematics, and is categorized as a sub-field of electrical engineering . Before the 1950s, signal processing was done in analogue, continuous-time systems like electronic circuits. Digital signal processing has grown in tandem with the use of digital computers and the microprocessor. Digital signal processing is in a sense more disconnected from electronics and physics than analogue signal processing, but even though the theory of signal processing has its roots in the very abstract and mathematical, the uses and applications are often very practical and physical. As this is a mathematical thesis, we will try to avoid the more physical components of signal processing and instead focus on the mathematical aspects. When this is said, signal processing is becoming more ”mainstream” for all engineers with the use of machine learning and other data-driven (in many cases signal-driven) methods.

This section is heavily inspired by *Principles of digital image processing* by W. Burger et. al [28], but we attempt to add some rigour and extra remarks to this work. A more detailed introduction to digital signal processing can be found in *Discrete-time signal processing* by Oppenheim et. al [18]. Other sources Fourier analysis and their use in signal

processing used are *Numerical analysis* by Sauer [19] and *Advanced engineering mathematics* by Kreyszig [14].

We will begin by mostly discussing so-called **continuous time** signals, where a signal $u(x)$ is a function of the continuous variable x. The word "continuous" can be somewhat confusing here as it refers to the variable x, not the continuity of the function $u(x)$. We will focus mostly on **discrete time** signals, where a signal u_k is a function of the discrete variable k. We will also discuss the process of turning a continuous time signal into a discrete time signal. The reason we discuss both continuous and discrete time signals is that this is essentially the difference between working in an infinite-dimensional and a finite-dimensional space.

We initially limit ourselves to one-dimensional signals, as the results easily generalize to two-dimensional signals, like images. We will mostly discuss real signals, for which some results simplify.

3.1 Fourier series and The Fourier Transform

Jean-Baptiste Joseph Fourier (1768-1830) was was a French mathematician, who spent much of his career researching heat flow and the partial differential equation called the heat equation.[21] He derived the heat equation, and found that its solution could be described by sine waves and cosine waves, **trigonometric** functions. The main insight of Fourier was that a general solution to the heat equation could be written using a possibly infinite sum of trigonometric functions, and expressing how to find the coefficients of this sum. This idea proved to be important in many fields of science, the industrial age to the information age.

We will use the complex exponential function

$$e^{ix} = \cos(x) + i\sin(x), \tag{3.1}$$

as it captures the properties of the $\cos$ and $\sin$ functions elegantly.

The main result of Fourier and other mathematicians was that any periodic function can be expressed as a sum of trigonometric functions, called a Fourier series. In fact, any integrable function defined on an interval $\mathbb{T}$ of length T such that $u : \mathbb{T} \to \mathbb{C}$ can be expressed as a Fourier series. The Fourier series is then a periodic function. The word "integrable" here means that u lies in the Lsbeque space

$$L_1(\mathbb{T}) = \{u(x) | \int_{\mathbb{T}} |u(x)|dx < \infty\}.$$

Definition 1. *(Fourier series) A function u that is integrable on an interval $\mathbb{T}$ of length T has a Fourier series*

$$v(x) = \sum_{k=0}^{\infty} A_k \cos(2\pi \frac{kx}{T}) + B_k \sin(2\pi \frac{kx}{T}), \tag{3.2}$$

where the coefficients A_k and B_k are the Fourier coefficients, given by

$$A_k = \frac{2}{T}\int_{\mathbb{T}} u(x)\cos(2\pi \frac{kx}{T})dx, \quad B_k = \frac{2}{T}\int_{\mathbb{T}} u(x)\sin(2\pi \frac{kx}{T})dx. \tag{3.3}$$

The integral is understood to be over a period of length T. If u is periodic with period T, the integral can be computed over any interval of length T. Alternatively, the function can be described by its complex Fourier series

$$v(x) = \sum_{k=-\infty}^{\infty} C_k e^{i2\pi \frac{kx}{T}}, \tag{3.4}$$

with complex Fourier coefficients

$$C_k = \frac{1}{T} \int_{\mathbb{T}} u(x) e^{-i2\pi \frac{kx}{T}} dx.. \tag{3.5}$$

We then have that $u(x) = v(x)$ almost everywhere.

A main result of Fourier analysis is that any integrable function on an interval T can be described by its Fourier series on that interval. This Fourier series is then a periodic function with period T.

Instead of period, one can also define the series using a fundamental frequency $\xi_0 = \frac{1}{T}$ or fundamental angular frequency $\omega_0 = \frac{2\pi}{T}$ The trigonometric functions used in the Fourier series have frequencies equal to integral multiples of the fundamental frequency, and these are called harmonic frequencies, or just harmonics.

The idea of the Fourier series can be extended by assuming the period is infinitely large, thus encapsulating functions that are integrable on $\mathbb{R}$. This is called the **Fourier transform**.

Definition 2. *The Fourier transform of a complex-valued, integrable function $u : \mathbb{C} \to \mathbb{C}$ is defined as*

$$\mathcal{F}(u(x)) = U(\xi) = \int_{-\infty}^{\infty} u(x) e^{-i2\pi\xi x} dx. \tag{3.6}$$

We will denote the Fourier transform of a signal with capital letters. The Fourier transform has many uses in electrical engineering and physics. In these fields x often represents time, and ξ often represents frequency. The Fourier transform translates a signal from its "time"-domain to its "frequency"-domain, also called the spectrum of the signal. We will in general call the space where u exists the **signal space** and the space of its spectrum the **Fourier space**. There exists many nearly equivalent conventions for the Fourier transform. For a short overview, see Appendix A.

A crucial mathematical property of the Fourier transform is its invertibility.

Definition 3. *(Inverse Fourier transform) For an integrable, complex-valued function $U : \mathbb{R} \to \mathbb{C}$, the inverse Fourier transform is defined as*

$$\mathcal{F}^{-1}(U(\xi)) = u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} U(\xi) e^{i2\pi\xi x} d\xi. \tag{3.7}$$

This is the inverse function of the Fourier transform.

The inverse Fourier transform is very similar in form to the Fourier transform. The invertibility property is also sometimes called the Fourier inversion theorem.

The Fourier transform has many other useful properties, which we will illustrate.

Lemma 2. *(Properties of the Fourier transform). Assume that u and v are complex-valued, integrable functions on $\mathbb{R}$. The Fourier transform then satisfies*

1. ***Linearity**. For scalars c_1 and c_2 we have that*

$$\mathcal{F}(c_1 u + c_2 v) = c_1 \mathcal{F}(u) + c_2 \mathcal{F}(v). \tag{3.8}$$

2. ***Conjugation**. The transform satisfies*

$$\mathcal{F}(u^*)(\xi) = \mathcal{F}(u)(-\xi)^*. \tag{3.9}$$

In particular, for purely real signals we have that

$$\mathcal{F}(u)(-\xi) = \mathcal{F}(u)(\xi)^* \iff |\mathcal{F}(u)(-\xi)| = |\mathcal{F}(u)(\xi)|. \tag{3.10}$$

3. ***Shift property**. Let $x' \in \mathbb{R}$, and $v(x) = u(x - x')$, then*

$$\mathcal{F}(v)(\xi) = e^{-i2\pi x'\xi}\mathcal{F}(u)(\xi). \tag{3.11}$$

Furthermore, let $\xi' \in \mathbb{C}$, and $v(x) = e^{i2\pi x\xi'}u(x)$, then

$$\mathcal{F}(v)(\xi) = \mathcal{F}(u)(\xi - \xi'). \tag{3.12}$$

4. ***Scaling property**. Let $c > 0$ be a scalar, then*

$$\mathcal{F}(u(cx))(\xi) = \frac{1}{c}\mathcal{F}(u(x))\left(\frac{\xi}{c}\right). \tag{3.13}$$

Proof. The proofs generally follow directly from the definition.
(1.) Follows directly from the linearity of the integral operator.
(2.) By the definition of the Fourier transform

$$\begin{aligned}
\mathcal{F}(u^*) &= \int_{-\infty}^{\infty} u^* e^{-i2\pi\xi x} dx \\
&= \int_{-\infty}^{\infty} \left(u e^{i2\pi\xi x}\right)^* dx \\
&= \left(\int_{-\infty}^{\infty} u e^{-i2\pi(-\xi)x} dx\right)^* = \mathcal{F}(u)(-\xi)^*.
\end{aligned}$$

The last line is only true because $dx^* = dx$, i.e we integrate over a real variable. Furthermore, if u is purely real then $u^* = u$. Thus we have

$$\mathcal{F}(\xi) = \mathcal{F}(u)(-\xi)^*. \tag{3.14}$$

Taking the absolute value on each side we obtain

$$|\mathcal{F}(u)(\xi)| = |\mathcal{F}(u)(\xi)|, \tag{3.15}$$

and the spectrum is symmetric around the origin.

(3.) Let $\xi' \in \mathbb{R}$ and $v(x) = u(x - x')$, then by the definition of the transform

$$\begin{aligned}\mathcal{F}(v) &= \int_{-\infty}^{\infty} u(x - x')e^{-i2\pi\xi x}dx \\ &= \int_{-\infty}^{\infty} u(y)e^{-i2\pi\xi(y+x')}dy \\ &= e^{-i2\pi x'}\int_{-\infty}^{\infty} u(y)e^{-i2\pi\xi y}dy \\ &= e^{-i2\pi x'}\mathcal{F}(u),\end{aligned}$$

using the change of variables $y = x - x'$. We have thus proven the first part of the shift property. Now, let $v(x) = u(x)e^{i2\pi x\xi'}$. Using the definition of the transform we have

$$\begin{aligned}\mathcal{F}(v) &= \int_{-\infty}^{\infty} u(x)e^{i2\pi x\xi'}e^{-i2\pi\xi x}dx \\ &= \int_{-\infty}^{\infty} u(x)e^{-i2\pi(\xi-\xi')x}dx \\ &= \mathcal{F}(u)(\xi - \xi').\end{aligned}$$

(4.) Let c be a real, non-zero scalar. Using the definition of the transform we have

$$\begin{aligned}\mathcal{F}(u(cx)) &= \int_{-\infty}^{\infty} u(cx)e^{-i2\pi\xi x}dx \\ &= \int_{-\infty}^{\infty} u(y)e^{-i2\pi\xi\frac{y}{c}}d\left(\frac{y}{c}\right) \\ &= \frac{1}{c}\mathcal{F}(u)\left(\frac{\xi}{c}\right),\end{aligned}$$

using the change of variables $y = cx$. □

The conjugation property is especially useful for purely real signals. In this case, the property means that the magnitude of the spectrum is symmetric around the origin.

3.1.1 Sampling

Continuous signals can be transformed to or interpreted as discrete signals through a process called **sampling**. We can interpret sampling in two ways: The discrete signal u_k sampled uniformly from a continuous signal $u(x)$ satisfies

$$u(kT_s) = u_k, \quad k \in Z, \tag{3.16}$$

where T_s is the **sampling period** and $Z \subset \mathbb{Z}$. Sampling might seem like an easy process, but it comes with some caveats. It is clear that the sampled signal does not contain all the information in the continuous signal.

The first concept we must introduce to understand this better are **convolutions**. Convolutions are the result of multiplying the spectrum of a function with another function, yet this is not immediately obvious.

Definition 4. *A convolution $u * v$ between two functions $u, v \in U$ returns a third function defined by*

$$(u * v)(x) = \int_{-\infty}^{\infty} u(y)v(x-y)dy. \tag{3.17}$$

The convolution can be thought of as measuring the overlap between two functions if we "pass" one function over the other. Convolutions are extremely useful in signal processing because of the property called the convolution theorem.

Theorem 1. *For two integrable functions $u, v \in U$, a convolution in signal space corresponds to multiplication in the Fourier space, i.e*

$$\mathcal{F}(u * v) = \mathcal{F}(u)\mathcal{F}(v). \tag{3.18}$$

Proof. Let u and v be integrable functions.

The Fourier transform of the convolution $u * v$ satisfies

$$\begin{aligned}\mathcal{F}(u * v) &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} u(y)v(x-y)dy\right] e^{-i2\pi\xi x}dx \\ &= \int_{-\infty}^{\infty} u(y)e^{-i2\pi\xi y}dy \int_{-\infty}^{\infty} v(z)e^{-i2\pi\xi z}dz \\ &= \mathcal{F}(u)\mathcal{F}(v).\end{aligned}$$

Here we substitute the variable $z = x - y$ and then separate the double integral into two integrals of separate variables. □

The convolution property is a marvelous property of the Fourier transform. It is worth noting that the exact same proof holds for the inverse transform. The convolution theorem holds both ways. Multiplication in signal space corresponds to a convolution in Fourier space, and vice versa.

In order to discuss sampling, we introduce the impulse or Dirac delta function, which also has close ties to the Fourier transform.

The impulse function can be loosely defined as a single impulse at a point, a function $\delta(x)$ satisfying

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & \text{otherwise} \end{cases}, \quad \int_{-\infty}^{\infty} \delta(x)dx = 1. \tag{3.19}$$

This definition is not particularly useful if rigour is the goal. Instead, we must define the δ function as a generalized function, or distribution. However, this is outside the scope of this text, and does not lead us towards the end goal of studying images. In this sense, all subsequent uses of the impulse functions are essentially abuse of notation, and have to be used with care. From now on, we also use a generalized Fourier transform, that allows for distributions.

Definition 5. *The impulse function δ is defined by the property that for a continuous function $u : \mathbb{R} \to \mathbb{C}$ we have that*

$$\int_{-\infty}^{\infty} u(x)\delta(x - x')dx = u(x'). \tag{3.20}$$

Moreover,

$$\int_{x'-\varepsilon}^{x'+\varepsilon} u(x)\delta(x-x')dx = u(x'). \tag{3.21}$$

for any $\varepsilon > 0$.

In other words, equation 3.20 lets us extract a single value $u(x')$ from the function $u(x)$, which is equvalent to sampling the point x'. We can thus interpret $u(x)\delta(x-x')$ as a sample of the signal $u(x)$ in the point x'. The next lemma illustrates some of the properties of the impulse function. The properties will be given without proof.

Lemma 3. *(Properties of the impulse function) Let* $u : \mathbb{R} \to \mathbb{C}$ *and* $x' \in \mathbb{C}$. *Then the following properties are satisfied:*

1. ***Fourier Transform***.

$$\mathcal{F}(\delta(x))(\xi) = 1. \tag{3.22}$$

Furthermore, let $\delta'(x) = \delta(x-x')$, *then*

$$\mathcal{F}(\delta'(x))(\xi) = e^{-i2\pi x'\xi}. \tag{3.23}$$

2. ***Convolution property***. *The impulse function is acts as the identity for convolution*

$$u(x) * \delta(x) = u(x), \tag{3.24}$$

or alternatively the shift operator

$$u(x) * \delta(x-x') = u(x-x'). \tag{3.25}$$

3. ***Scaling property***. *Let* c *be a scalar, then*

$$\int_{-\infty}^{\infty} u(x)\delta(cx) = {}_{-\infty}^{\infty} u(x)\frac{\delta(x)}{|c|}. \tag{3.26}$$

4. ***Symmetry***.

$$\delta(-x) = \delta(x). \tag{3.27}$$

The properties of the impulse function should be reminiscent of the properties of the Fourier transform, and we will see that these two ideas are linked. The third property is more often written as

$$\delta(cx) = \frac{\delta(x)}{|c|}, \tag{3.28}$$

which is an abuse of notation. We will use this notation from now on.

We can extend the idea of the impulse function to "sample" infinitely many discrete values from a signal. We will only consider uniform sampling, where one samples equidistant points. We call the distance between these points the sampling period $T_s > 0$, with corresponding sampling rate $\xi_s = \frac{1}{T_s}$. To formalize this, we introduce the so called Dirac comb function, also called the **sampling function**.

Definition 6. *(Sampling function) The sampling function W_{T_s} with sampling rate $\xi_s = \frac{1}{T_s}$ is defined as*

$$W_{T_s}(x) = \sum_{k=-\infty}^{\infty} \delta(x - kT_s) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} \delta(\frac{x}{T_s} - k) = \frac{1}{T_s} W_1(\frac{x}{T_s}). \tag{3.29}$$

From lemma 3, some properties of the sampling function are immediately apparent. We can interpret $u(x)W_{T_s}(x)$ as infinitely many equidistant samples of the $u(x)$. The resulting function is infinitely many impulses where each impulse is weighted with the value of $u(x)$ in that point. In this sense, we can extract the value of the sample by the operation

$$u(kT_s) = u_k = \int_{kT_s-\varepsilon}^{kT_s+\varepsilon} u(x) \cdot W_{T_s}(x) dx, \tag{3.30}$$

for any $0 < \varepsilon < T_s$. This might seem cumbersome. If we wanted to interpret sampling in continous time, a choice of interpretation is

$$\tilde{u}(x) = \begin{cases} u_k, & x = kT \\ 0, & \text{otherwise} \end{cases} \tag{3.31}$$

The problem with a function like this is that any integral will be 0. In this sense, none of the points have any "weight", which does not translate well into actual discrete signals where each point has a weight equal to u_k. This means that a function like $\tilde{u}$ has Fourier transform equal to 0, which means we cannot use the tools from Fourier analysis. Thus we interpret the sampling of a function $u(x)$ as $u(x)W_{T_s}(x)$.

The sampling function lets us define an important property of periodic functions, namely **periodic summation**.

Definition 7. *(Periodic summation) For a function $u : \mathbb{R} \to \mathbb{C}$, the periodic summation of u with period T, denoted by u_T is defined as*

$$u_T = \sum_{k=-\infty}^{\infty} u(x - kT) = u * W_T \tag{3.32}$$

Without getting too technical, the periodic summation does not always exist for all functions u. The function u_T is obviously periodic. In this sense we can create periodic functions out of aperiodic functions.

With the idea of periodic summation, we can introduce a crucial relationship between continuous time and discrete time Fourier transforms.

Lemma 4. *(Poisson summation formula) Assume the periodic summation u_T converges uniformly. We then have*

$$u_T = \sum_{k=-\infty}^{\infty} u(x - kT) = \frac{1}{T} \sum_{k=-\infty}^{\infty} \mathcal{F}(u(x))(\frac{k}{T}) e^{i2\pi \frac{kx}{T}}. \tag{3.33}$$

Furthermore,

$$\sum_{k=-\infty}^{\infty} u(k) = \sum_{k=-\infty}^{\infty} U(k). \tag{3.34}$$

Proof. The periodic summation u_T is periodic with period T. The Fourier coefficients are

$$\begin{aligned} C_k &= \frac{1}{T}\int_0^T [\sum_{k=-\infty}^{\infty} u(x+kT)]e^{-i2\pi\frac{kx}{T}}dx \\ &= \frac{1}{T}\sum_{k=-\infty}^{\infty}\int_0^T u(x+kT)e^{-i2\pi\frac{kx}{T}}dx \\ &= \frac{1}{T}\sum_{k=-\infty}^{\infty}\int_k^{k+T} u(x)e^{-i2\pi\frac{kx}{T}}dx \\ &= \frac{1}{T}\int_{-\infty}^{\infty} u(x)e^{-i2\pi\frac{kx}{T}}dx \\ &= \frac{1}{T}\mathcal{F}(u)(\frac{k}{T}) = \mathcal{F}(u(xT))(k) \end{aligned}$$

Thus, u_T can be written as a Fourier series.

$$u_T = \frac{1}{T}\sum_{k=-\infty}^{\infty}\mathcal{F}(u(x))(\frac{k}{T})e^{i2\pi\frac{kx}{T}}. \tag{3.35}$$

Inserting $x = 0$ and $T = 1$ yields

$$\sum_{k=-\infty}^{\infty} u(k) = \sum_{k=-\infty}^{\infty} U(k). \tag{3.36}$$

□

The main result of the Poisson summation formula is that the periodic summation of a function $u(x)$ can be written as a Fourier series where the coefficients are equal to samples of the Fourier transform of $u(x)$. By the duality of the Fourier transform, we can instead write the Poisson summation as

$$U_T = \sum_{k=-\infty}^{\infty} U(\xi - \frac{k}{T}) = Tu(kT)e^{i2\pi\xi kT}. \tag{3.37}$$

This is perhaps easier understood by realizing that the Fourier transform of the sampling function is another sampling function, which we will show in the next lemma.

Lemma 5. *The sampling function W_{T_s} can be written as a Fourier series*

$$W_{T_s}(x) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} e^{i2\pi\frac{kx}{T_s}}. \tag{3.38}$$

Furthermore, the Fourier transform of the sampling function is another sampling function

$$\mathcal{F}(W_{T_s}(x))(\xi) = \frac{1}{T_s}W_{1/T_s}(\xi) = \sum_{k=-\infty}^{\infty}\delta(\frac{x}{T_s} - k) = \sum_{k=-\infty}^{\infty} e^{-i2\pi kxT_s} \tag{3.39}$$

Proof. Inserting $u(x) = \delta(x)$ into lemma 4 immediately yields that

$$W_{T_s}(x) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} e^{i2\pi \frac{kx}{T_s}}. \tag{3.40}$$

For the Fourier transform, we first use the scaling property of the Fourier transform

$$\mathcal{F}(W_{T_s}(x))(\xi) = \mathcal{F}(\frac{1}{T_s} W_1(\frac{x}{T_s})) = \mathcal{F}(W_1(x))(T_s\xi), \tag{3.41}$$

and using the definition of the Fourier transform this becomes

$$= \int_{-\infty}^{\infty} [\sum_{k=-\infty}^{\infty} \delta(x-k)] e^{i2\pi\xi x T_s} dx = \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(x-k) e^{i2\pi\xi x T_s} dx. \tag{3.42}$$

We recognize this as the integral shift property of the impulse function, and this becomes

$$\sum_{k=-\infty}^{\infty} e^{i2\pi\xi k T_s}, \tag{3.43}$$

which is a Fourier series. Using the previous Fourier series result we find that this Fourier series corresponds to

$$\sum_{k=-\infty}^{\infty} e^{i2\pi\xi k T_s} = \frac{1}{T_s} W_{1/T_s}(\xi). \tag{3.44}$$

which concludes the proof. □

We can now reinvestigate the poisson summation formula, and ask ourselves: What is the Fourier transform of a sampled signal? It should be clear by now that

$$\mathcal{F}(uW_{T_s}) = \frac{1}{T_s} U(\xi) * W_{1/T_s} = \frac{1}{T_s} U_{1/T_s}. \tag{3.45}$$

In other words, the spectrum of a sampled function is a periodic summation of the spectrum of the function. In order to properly introduce what this means, we must talk about the **bandwidth** of a signal. The bandwidth of a signal is equal to the distance between the maximal frequency ξ_{max} and the minimal non-negative frequency ξ_{min}. ξ_{max} is the largest ξ where the power spectrum $|U(\xi)|$ has support. For many real signals, ξ_{min} is usually 0. In this case the bandwidth is the same as ξ_{max}. In this case the support of the spectrum is in $[-\xi_{\text{max}}, \xi_{\text{max}}]$. When sampling from a signal like this, there can be problems. If the sampling frequency $\xi_s = 1/T_s$ is small relative to ξ_{max}, the periods periodic summation of the spectrum can "bleed" into each other. This is a phenomenon called **aliasing**, and is illustrated in figure 3.1.

What aliasing means is that we cannot extract the continuous signal from a sampled signal if it has been sampled with a low sampling frequency compared to the bandwidth of the signal. This means that different signals can become indistinguishable (aliases of one another) when sampled.

We have basically shown the Nyquist-Shannon theorem, which we will now stated as Shannon originally stated it.

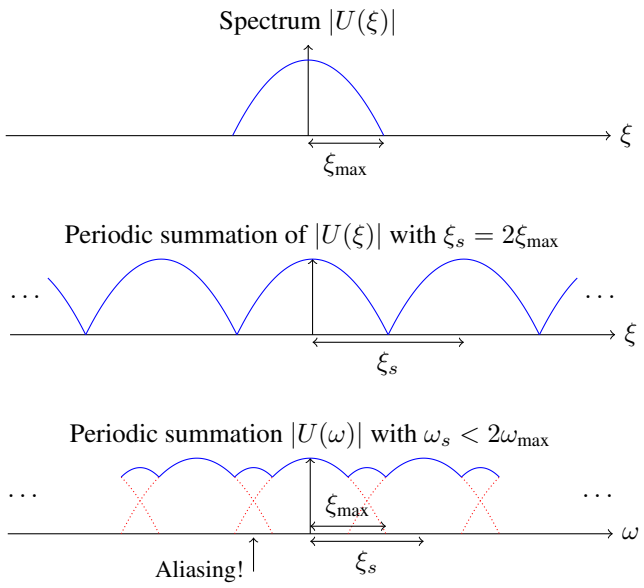

Figure 3.1: Effects on the spectrum $U(\xi)$ when sampling from a signal $u(x)$. In the first figure we see the spectrum of the original continuous signal. In the second figure we see the signal samples with "perfect" sampling frequency $\xi_s = 2\xi_{\max}$. The spectrum is periodic, and perfectly captures all the information of the original signal. In the last figure we see the same but with $\xi_s < 2\xi_{\max}$. The periods "bleed" into each other, and the resulting spectrum does not represent the original continuous signal.

Theorem 2. *(Nyquist-Shannon) If a signal $u(x)$ has no frequencies higher than ξ_{max}, it is completely determined by giving its ordinates at a series of points spaced $\frac{1}{2\xi_{max}}$ apart.*

Another way of stating this theorem is that in order to avoid aliasing, we need the sampling frequency to satisfy

$$\frac{\xi_s}{2} \geq \xi_{\max}. \tag{3.46}$$

This is the sampling theorem, and is essential when sampling signals. The quantity $\frac{\xi_s}{2}$ is usually called the **Nyquist frequency**.

The act of Fourier transforming a sampled signal also goes by another name, the Discrete Time Fourier Transform (DTFT).

Definition 8. *(Discrete Time Fourier Transform) The Discrete Time Fourier Transform (DTFT) of a discrete signal $u_k = u(kT)$ is defined as*

$$DTFT_{1/T}(u)(\xi) = \sum_{k=-\infty}^{\infty} Tu(kT)e^{-i2\pi k\xi T} = \sum_{k=-\infty}^{\infty} U(\xi - \frac{k}{T}) = \mathcal{F}(\sum_{k=-\infty}^{\infty} u(kT)\delta(x-kT)). \tag{3.47}$$

This definition summarizes periodic summation, poisson summation and sampling. Sampling in one space corresponds to periodic summation in the other.

The DTFT of a discrete signal is a function, which is not easy to handle on a computer. We would like something akin to the Fourier transform that also returns a discrete signal.

3.1.2 The discrete Fourier Transform

We can finally introduce the **Discrete Fourier Transform** (DFT).

Definition 9. *(Discrete Fourier Transform) The Discrete Fourier Transform (DFT) of a discrete, summable, signal u with elements u_k, $k = 0, ...n-1$ is defined as*

$$DFT(u)_l = U_l = \sum_{k=0}^{n-1} u_k e^{-i2\pi \frac{lk}{n}}. \tag{3.48}$$

The result of the DFT is a new discrete signal U with elements U_l, $l = 0, ...n-1$.

The DFT is discrete analog to the Fourier coefficients of a signal, not the Fourier transform as the name implies. If we calculate the $l-th$ Fourier coefficients of a signal $u(x)$ sampled n times over a period T s we obtain

$$\begin{aligned} C_l &= \frac{1}{T}\int_0^T \sum_{k=0}^{n-1} u(kT)\delta(x-kT)e^{-i2\pi\frac{lx}{T}}dx \\ &= \sum_{k=0}^{n-1} u(kT)\int_0^T \delta(x-kT)e^{-i2\pi\frac{lx}{T}}dx \\ &= \sum_{k=0}^{n-1} u(kT)e^{-i2\pi\frac{kl}{n}}, \end{aligned}$$

Which is exactly the DFT. Note the Fourier coefficients C_l are periodic with period n in this case, which is consistent with what we have concluded earlier in this text.

The DFT is also motivated by the DTFT. If we assume a discrete signal $u(kT)$ of period n, the DTFT should converge to zero everywhere except for integer multiples of $1/nT$, the harmonic frequecnies of u. We can see this by rearranging the DTFT to obtain

$$\begin{aligned} DTFT_{1/T}(u)(\xi) &= \sum_{k=-\infty}^{\infty} u_k e^{-i2\pi k\xi T} \\ &= \sum_{l=-\infty}^{\infty}\sum_{k=0}^{n-1} u_{nl+k}e^{-i2\pi(nl+k)\xi T} \end{aligned}$$

We see that this sum diverges to infinity for integer multiples of $\xi = \frac{1}{nT}$ at the rate and goes to 0 for any other ξ. Here we use the notion that the infinite sum of something periodic

is 0. If we insert an integer multiple $\xi = \frac{m}{nT}$ we obtain

$$\begin{aligned} DTFT_{1/T}(u)(m/nT) &= \sum_{l=-\infty}^{\infty} \sum_{k=0}^{n-1} u_k e^{-i2\pi \frac{(nl+k)m}{n}} \\ &= \sum_{l=-\infty}^{\infty} [\sum_{k=0}^{n-1} u_k e^{-i2\pi \frac{km}{n}}] e^{-i2\pi lm} \\ &= DFT(u)_m \sum_{l=-\infty}^{\infty} 1 \end{aligned}$$

In other words, it diverges with the rate of the DFT coefficients. We can use this to assert that the DTFT of a periodic discrete signal is perfectly described by the DFT coefficients. What we have just done is essentially to sample n points of the DTFT. If we now assume a finite discrete signal $u(kT)$ of length n, and we want to sample n points of the DTFT we obtain

$$DTFT_{1/T}(u)(l/nT) = \sum_{k=-\infty}^{\infty} u_k e^{-i2\pi k(\frac{l}{nT}T} = \sum_{k=0}^{n-1} u_k e^{-i2\pi \frac{lk}{n}}, \tag{3.49}$$

which is exactly the DFT. It should be noted that the DFT can suffer from aliasing, and it is important to remember that the DFT is periodic. The bandwidth of a discrete signal is always infinite, but the spectrum is periodic. Thus the bandwidth of any interest is "squeezed" inside the first period of the DTFT, which we can sample from using the DFT. This period depends on the size n of the signal. We can interpret the sampling theorem from a different perspective. While a discrete signal of length n can attain infinitely many values, it can only capture a finite "amount of frequency".

We have so far introduced four different tools of Fourier analysis: Fourier series, Fourier transform, DTFT and DFT. A depiction of these tools is shown in figure 3.2.

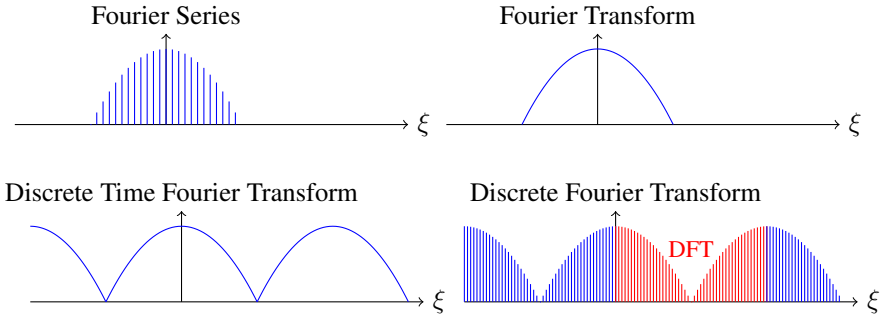

Figure 3.2: The different Fourier transforms.

The DFT has many of the properties analogous to the continuous Fourier transform. Firstly, it has an inverse.

Definition 10. *(Inverse DFT) The inverse DFT of a discrete signal U is defined as*

$$DFT^{-1}(U)_k = u_k = \frac{1}{n}\sum_{l=0}^{n-1} U_l e^{i2\pi\frac{lk}{n}}, \quad k = 0, ..., n-1. \tag{3.50}$$

We can restate the properties of the continous Fourier transform in lemma 2 in the discrete case.

Lemma 6. *(Properties of the* DFT*)*

1. ***Linearity***.
$$\mathrm{DFT}(c_1 u + c_2 v) = c_1 \mathrm{DFT}(u) + c_2 \mathrm{DFT}(v). \tag{3.51}$$
2. ***Conjugation***.
$$\mathrm{DFT}(u^*)_l = \mathrm{DFT}(u_{-k})^*_l \tag{3.52}$$
3. ***Circular shift property***
$$\mathrm{DFT}(u \cdot e^{i2\pi\frac{ll'}{n}})_l = \mathrm{DFT}(u)_{l-l'}. \tag{3.53}$$

Proofs are analogous to the continuous case.

Remembering that the DFT is periodic with period n, we use the notation $u_{-k} = u_{n-k}$. This means that the shift property should be interpreted as a circular shift. We can also describe a circular shift as shifted modulo N.

We can now discuss **discrete convolutions**, which can be defined identically to continuous convolutions.

Definition 11. *(Discrete linear convolution) For discrete signals u and v, the discrete convolution $u * v$ is defined as*

$$(u * v)_l = \sum_{k=-\infty}^{\infty} u_l v_{l-k}. \tag{3.54}$$

We use the word "linear convolution" to separate this convolution from the different convolutions we will show later.

For finite discrete signals, the support of the convolution (or just the size of the resulting vector) depends on the size n of u and m of v. For the purpose of simplicity, we will always assume that n is an even number while m is an odd number. Here the convolution can be defined as

$$(u * v)_l = \sum_{k=-(m+1)/2}^{n+(m+1)/2} u_l v_{l-k}. \tag{3.55}$$

The result is a new signal of size $m + n - 1$.

We can now investigate convolution theorems similar to the continuous case. The goal is to calculate the linear convolution. The original convolution theorem obviously holds for the DTFT, as this is just a Fourier transform applied to discrete signals.

$$DTFT_{1/T}(u * v) = DTFT_{1/T}(u) DTFT_{1/T}(v). \tag{3.56}$$

In order to find a convolution theorem for the DFT, we can essentially apply the proof for the original convolution theorem in reverse. What is

$$\mathrm{DFT}^{-1}(\mathrm{DFT}(u)\,\mathrm{DFT}(v))? \tag{3.57}$$

This depends on the properties of the signals u and v. Firstly, we assume the signals are periodic with period n. We then arrive at the idea of a periodic convolution, which can be defined as

$$(\tilde{u} * \tilde{v})_k = \sum_{l=0}^{n-1} \tilde{u}_l \tilde{v}_{k-l}. \tag{3.58}$$

It should be noted that the periodic convolution is defined over just one period, but is periodic with period n. A linear convolution of two infinite signals almost always diverges.

For periodic convolutions, i.e linear convolutions of periodic signals, we have that

$$\mathrm{DFT}(\tilde{u} * \tilde{v}) = \mathrm{DFT}(\tilde{u})\,\mathrm{DFT}(\tilde{v}), \tag{3.59}$$

which is the convolution theorem! This can be shown using the exact same proof as for the continuous case and using the circular shift property of the DFT.

What if we want to use the DFT to calculate the convolution between two finite (non-periodic) signals? We can turn the finite signals into periodic signals using periodic extensions discussed earlier. If we define u to be one period of $\tilde{u}$, i.e

$$\tilde{u}_k = \tilde{u} * W_n. \tag{3.60}$$

The shift of a periodic signal of period n is a circular shift, which means that for $0 \leq l, k \leq n-1$ we have

$$\tilde{u}_{l-k} = \begin{cases} u_{l-k}, & l-k \geq 0 \\ u_{n-(k-l)}, & l-k < 0. \end{cases} \tag{3.61}$$

If we insert this into the periodic convolution, we obtain

$$(\tilde{u} * \tilde{v})_k = \sum_{l=0}^{n-1} \tilde{u}_l \tilde{v}_{l-k} = \sum_{l=0}^{n-1} u_l \tilde{v}_{l-k}. \tag{3.62}$$

The second equality is because $\tilde{u}_l = u_l$ for $0 \leq l \leq n-1$. We call this kind of convolution a **Circular convolution**. If we apply the convolution theorem for periodic sequences we see that

$$\mathrm{DFT}(\tilde{u} * \tilde{v}) = \mathrm{DFT}(u * \tilde{v}) = \mathrm{DFT}(u)\,\mathrm{DFT}(\tilde{v}), \tag{3.63}$$

where the tilde denotes a periodic extention of an aperiodic signal.

We can use a circular convolution to calculate a linear convolution with aliasing. When multiplying the discrete Fourier transforms of two signals finite signals of different length, we must pad the signals to be of the same length.

We will see that the act of padding a signal with zeros gives us a DFT with higher "resolution", and is actually the same as interpolating the DFT. For a signal u with length n, a zero padded signal of length $n+m$ satisfies

$$P_m(u)_k = \begin{cases} u_k, & k \leq n-1 \\ 0, & \text{otherwise} \end{cases}, \tag{3.64}$$

where P_m is an operator that pads the signal with m zeroes.

The DFT of a padded signal satisfies

$$\mathrm{DFT}(P_m(u))_l = \sum_{k=0}^{n} u_k e^{-i2\pi \frac{lk}{n+m}}, \tag{3.65}$$

which is a signal of length $n + m$. We recognize this as a finer sampling of the DTFT. While $\mathrm{DFT}(u)$ consists of n samples of the DTFT over one period, $\mathrm{DFT}(P_m(u))$ consists of $n + m$ samples of the DTFT over one period. This corresponds to a specific upsampling of $\mathrm{DFT}(u)$. This is not the same as using a finer sampling of a continuous signal, and cannot on its own be used to reduce aliasing. In order to apply a circular convolution to two signals u of length n and v of length m, we must pad the two signals to be of the same length.

If we pad the signals to $\max(n, m)$, or just assume they have the same length, we achieve a linear convolution with aliasing. The argument is essentially the same as for why sampling signals can yield aliasing, only in Fourier space. Because the linear convolution between two signals of length n and m is of length $n + m - 1$, we need at least $n + m - 1$ DFT coefficients to properly describe the signal. If not, the periods of the convolution in signal space will overlap, and aliasing will occur.

To summarize, a linear convolution of two signals of length n and m can be calculated as a circular convolution with aliasaing. In order to avoid aliasing, we need to pad both signals to length $n + m - 1$. This will become more clear in later sections.

3.1.3 Vector perspective of the DFT

We will now approach the DFT from a more mathematical perspective. This allows for a deeper understanding as well as makes it easier to show some needed results.

We now interpret a discrete signal u as a vector in $\mathbb{R}^n$ of length n, whose elements we denote u_k. Any linearly independent set of n vectors of size n is a basis for this space. A particularily interesting orthogonal basis are the normalized version of the vectors used in the DFT

$$(q_k)_l = \frac{1}{\sqrt{n}} e^{12\pi \frac{kl}{n}}. \tag{3.66}$$

The corresponding basis functions in infinite dimensions are the functions

$$q_k = \frac{1}{T} e^{i2\pi \frac{kx}{T}}, \tag{3.67}$$

which also form an orthogonal basis for the infinite-dimensional space $L^2(\mathbb{R})$. We can easily see that

$$\sum_{l=0}^{n-1} (q_k)_l = \begin{cases} 1, & k = 0 \\ 0, & k > 0. \end{cases} \tag{3.68}$$

This can be easily seen by recognizing this as a geometric series. Let $\theta = (q_1)_0 = \frac{1}{n} e^{i2\pi \frac{l}{T}}$, which satisfies $\theta^n = 0$, then

$$\sum_{l=0}^{n-1} (q_k)_l = 1 + \theta + \cdots + \theta^{n-1} = \frac{\theta^n - 1}{\theta - 1} = 0. \tag{3.69}$$

A direct consequence of this is that

$$\langle q_k, q_{k'} \rangle = \frac{1}{n} \sum_{l=0}^{n-1} e^{i2\pi \frac{(k-k')l}{n}} = \begin{cases} 1, & k = k' \\ 0, & \text{otherwise} \end{cases} = \delta_{kk'}, \tag{3.70}$$

and the basis functions are orthonormal. Here we also defined the Kroenecker delta function $\delta_{kk'}$, which is the discrete analogue of the Dirac delta function.

Using the basis vectors we can alternatively write each element of the DFT as the inner product

$$\mathrm{DFT}(u)_k = \sum_{l=0}^{n-1} u_l (e^{i2\pi \frac{kl}{n}})^* = n \langle u, q_k \rangle. \tag{3.71}$$

Furtheremore, the inverse DFT can be written as a linear combination of the Fourier basis vectors

$$\mathrm{DFT}^{-1}(U) = \sum_{k=0}^{n-1} U_k \bar{q}_k. \tag{3.72}$$

Both the DFT and inverse can then be written as a matrix operation. The DFT can be written as a matrix multiplication with the so-called **Fourier matrix**

$$Q_n = \begin{bmatrix} (q_0)_0 & \cdots & (q_{n-1})_0 \\ \vdots & & \vdots \\ (q_0)_{n-1} & \cdots & (q_{n-1})_{n-1} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \cdots \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & \cdots \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \cdots \omega^{(n-1)^2} \end{bmatrix}, \tag{3.73}$$

where ω is the n-th root of unity. We see that the Fourier matrix is symmetric, and the inverse is the complex conjugate

$$Q_n^{-1} = \bar{Q_n} = \begin{bmatrix} \omega^0 & \omega^0 & \cdots \omega^0 & \\ \omega^0 & \omega^{-1} & \omega^{-2} & \cdots \omega^{-(n-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \cdots \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \vdots \\ \omega^0 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots \omega^{-(n-1)^2} \end{bmatrix}, \tag{3.74}$$

which is also symmetric. The Fourier matrix is thus orthogonal, as $Q_n Q_n^* = I$. We can write the forward and inverse Discrete Fourier transforms as matrix operations

$$\mathrm{DFT}(u) = nQ_n u, \quad \mathrm{DFT}^{-1}(U) = Q_n^* U. \tag{3.75}$$

A useful relationship between a signal u and its DFT is Parseval's theorem.

Theorem 3. *(Parseval's theorem) Let $u, v \in \mathbb{R}^n$, then*

$$\langle u, v \rangle = \frac{1}{n} \langle \mathrm{DFT}(u), \mathrm{DFT}(v) \rangle. \tag{3.76}$$

Furthermore,

$$\langle u, v\rangle = \sum_{k=0}^{n-1} \langle u, q_k\rangle \langle v, q_k\rangle^* \tag{3.77}$$

For a differently scaled DFT, this implies that the DFT is a unitary operator.

We can deduce what the different elements of the DFT of a discrete signal represent. The wave number k is a measure of how many cycles the Fourier basis function makes over the signal length n. For $k = 0$, the basis function is just a vector containing 1. In other words, the real part of the zeroth entry of $\mathrm{DFT}(u)$ contains the constant part of the signal, or the average of the signal.

We can now define a linear convolution as a linear operator

$$Au = \kappa * u, \tag{3.78}$$

for some kernel κ of size m. From the previous section, we found that

$$Au = \mathrm{DFT}^{-1}(\mathrm{DFT}(\kappa)\,\mathrm{DFT}(u)), \tag{3.79}$$

where the DFT is of size $n+m-1$. We can combine this with equations (3.71) and (3.72) to find

$$Au = \sum_{k=0}^{n+m-1} \underbrace{\mathrm{DFT}(\kappa)_k \langle u, q_k\rangle}_{(\mathrm{DFT}(\kappa)\,\mathrm{DFT}(u))_k} \underbrace{q_k}_{\mathrm{DFT}^{-1}}. \tag{3.80}$$

Using this formulation of convolutions, we see that

$$Aq_k = \mathrm{DFT}(\kappa)_k q_k. \tag{3.81}$$

For all convolution kernels κ, the eigenvectors of the convolution operator A is the Fourier basis vectors q_k, and the eigenvalues are the DFT coefficients of the kernel κ. Thus, we can write a convolution as a matrix diagonalization

$$Au = Q_{n+m-1}^* \,\mathrm{diag}(\mathrm{DFT}(\kappa)) Q_{n+m-1} u, \tag{3.82}$$

where $\mathrm{diag}(\mathrm{DFT}(\kappa))$ is a diagonal matrix containing the DFT coefficients of κ.

The matrix diagonalization is an absolutely beautiful reformulation of the convolution theorem. We can interpret the matrix diagonalization in three parts:

- Q_{n+m-1} corresponds to the DFT.
- $\mathrm{diag}(\mathrm{DFT}(\kappa))$ corresponds to element-wise multiplication with the DFT of another signal.
- Q_{n+m-1}^* corresponds to the inverse DFT.

We can use this to investigate the inverse operator A^{-1}. It can be written as

$$A^{-1}u = Q_{n+m-1}^* \,\mathrm{diag}(\mathrm{DFT}(\kappa))^{-1} Q_{n+m-1} u = \mathrm{DFT}^{-1}\left(\frac{\mathrm{DFT}(u)}{\mathrm{DFT}(\kappa)}\right), \tag{3.83}$$

where the division is interpreted element-wise. This means that the inverse of a convolution operator only exists is all DFT coefficients are non-zero. We can also use this to find the condition number of the operator A,

$$\mathcal{K}(A) = \frac{\max_l |\operatorname{DFT}(\kappa)_l|}{\min_l |\operatorname{DFT}(\kappa)_l|}. \tag{3.84}$$

A convolution operator is ill-conditioned if there is a large span in the DFT coefficients of the kernel κ.

Two other operators we will need later are A^* and A^*A. We can find A^* by applying Parsevals theorem

$$\begin{aligned}
\langle Au, v\rangle &= \frac{1}{n+m-1}\langle \operatorname{DFT}(Au), \operatorname{DFT}(v)\rangle = \langle \operatorname{DFT}(\kappa)\operatorname{DFT}(u), \operatorname{DFT}(v)\rangle \\
&= \frac{1}{n+m-1}\sum_{l=0}^{n+m-1} \operatorname{DFT}(\kappa)_l \operatorname{DFT}(u)_l \operatorname{DFT}(v)_l^* \\
&= \frac{1}{n+m-1}\sum_{l=0}^{n+m-1} \operatorname{DFT}(\kappa)_l \underbrace{(\operatorname{DFT}(u)_l^* \operatorname{DFT}(v)_l)^*}_{\operatorname{DFT}(A^*v)_l} \\
&= \langle u, A^*v\rangle.
\end{aligned}$$

We find that

$$A^*u = \operatorname{DFT}^{-1}(\operatorname{DFT}(\kappa)^* \operatorname{DFT}(u)). \tag{3.85}$$

We can now apply the convolution theorem and the conjugation property of the DFT to find that A^* is another convolution operator satisfying

$$(A^*v)_k = \sum_{l=0}^{n+m-1} \kappa_{l-k}^* v_l. \tag{3.86}$$

The adjoint of a convolution operator is equivalent to a convolution with the kernel κ conjugated and flipped, so the elements are in reversed order.

As for the operator A^*A, we find that this is equal to

$$\begin{aligned}
A^*Au &= \sum_{k=0}^{n+m-1} \operatorname{DFT}(\kappa)_k^* \langle Q_{n+m-1}^* \operatorname{diag}(\operatorname{DFT}(\kappa)) Q_{n+m-1} u, q_k\rangle q_k \\
&= \sum_{k=0}^{n+m-1} \operatorname{DFT}(\kappa)_k^* \langle \operatorname{diag}(\operatorname{DFT}(\kappa)) Q_{n+m-1} u, Q_{n+m-1} q_k\rangle q_k \\
&= \sum_{k=0}^{n+m-1} |\operatorname{DFT}(\kappa)_k|^2 \langle u, q_k\rangle q_k \\
&= \operatorname{DFT}^{-1}(|\operatorname{DFT}(\kappa)|^2 \cdot \operatorname{DFT}(u)).
\end{aligned}$$

3.1.4 The Discrete Fourier Transform in 2D

For image applications, we need to extend the DFT to 2D. Essentially all we have shown in the previous sections also apply to the 2D case with extra indices.

For a 2D, discrete signal u_{k_x,k_y}, with $k_x = 0, ..., n-1$ and $k_y = 0, ..., n-1$, the DFT can be defined as

$$U_{l_x,l_y} = \sum_{k_x=0}^{n-1} \sum_{k_y=0}^{n-1} u_{k_x,k_y} e^{-i2\pi\left(\frac{k_x l_x}{n} + \frac{k_y l_y}{n}\right)}, \quad l_x = 0, ...n-1, l_y = 0, ..., n-1. \tag{3.87}$$

The 2D DFT can be decoupled into multiple 1D DFTs

$$U_{l_x,l_y} = \sum_{k_x=0}^{n-1} \underbrace{\left[\sum_{k_y=0}^{n-1} u_{k_x,k_y} e^{-i2\pi \frac{k_y l_y}{n}}\right]}_{\text{1D DFT}} e^{-i2\pi \frac{k_x l_x}{n}}. \tag{3.88}$$

In other words, a 2D DFT is as computationally expensive as applying n^2 1D DFTs.

The result from the DFT is a $n \times n$ discrete signal, which can be interpreted as periodic in both dimensions

$$U_{l_x+c_x n, l_y+c_y n} = U_{l_x,l_y} \tag{3.89}$$

for any integers c_x and c_y.

In order to visualize the 2D DFT, one usually works with the magnitude of the DFT, $|U_{l_x,l_y}|$, also called the **power spectrum** of the signal. As in one dimension, the power spectrum of a real signal (images) is symmetric about the origin, i.e

$$|U_{l_x,l_y}| = |U - l_x, -l_y|. \tag{3.90}$$

Because of this, it is common to swap the four quadrants of the power spectrum in order to obtain a so-called **centered representation** when visualising. This means that the low-frequency coefficients are found in the center of the image and the high-frequency coefficients are found near the edges of the image. Finally, because the coefficients usually span a large range, it is common to plot the logarithm of the power spectrum $\log |U_{l_x,l_y}|$. Examples of the 2D DFT is shown in figure 3.3.

From the figure we see the usefulness of the centered DFT. The second signal shown, the brick wall, is actually a periodic image. Because of this, we see the DFT tends towards discrete frequencies. In both signals, and particularly the first, the frequencies in the x and y direction are large compared to the other frequencies. This is partly because of the periodic extension of the original signal. An illustration of the periodic extension of the two signals is shown in figure 3.4. In the first image, the periodic extension creates some very abrupt changes in the x and y direction. We see this in the DFT coefficients, as most of the power spectrum is in the x and y direction.

3.1.5 2D discrete convolution

Linear discrete convolutions in 2D can be defined as

$$(u * v)_{k_x,k_y} = \sum_{l_x=-\infty}^{\infty} \sum_{l_y=-\infty}^{\infty} u_{l_x,l_y} v_{k_x-l_x,k_y-l_y}. \tag{3.91}$$

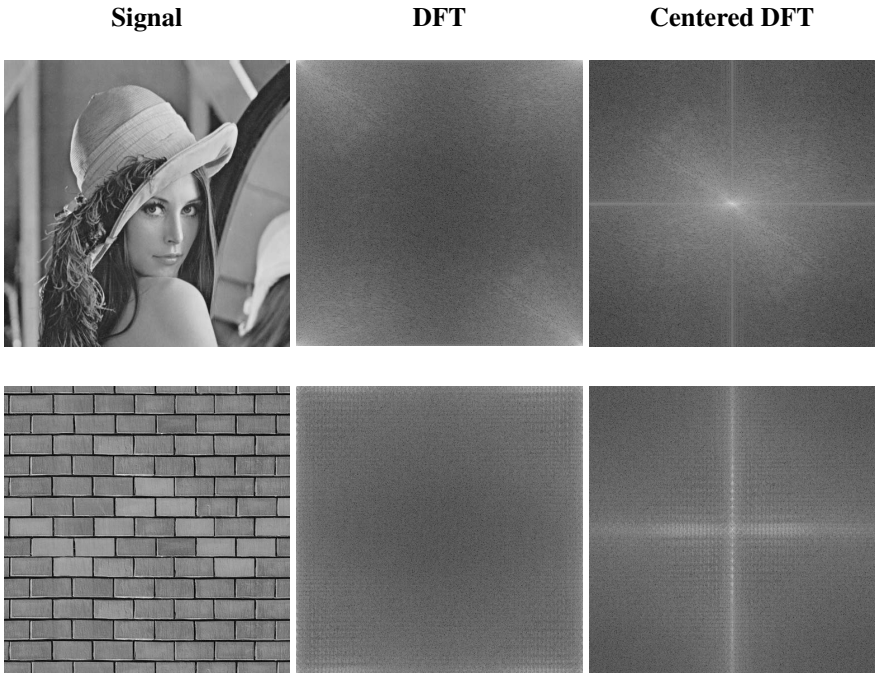

Figure 3.3: Two signals and the logarithm of the power spectrum $\log|U|$ using normal DFT and centered DFT. In the centered DFT, the low frequencies are in the center, and the high frequencies are closer to the edges.

Figure 3.4: Periodic extenstions of the signals in figure 3.3.

We will now attempt to show how the circular convolution theorem can be used to calculate linear discrete convolutions. Figure 3.5 shows the convolution procedure between an image u with resolution $n \times n$ and a kernel κ with resolution $m \times m$. Here the kernel is

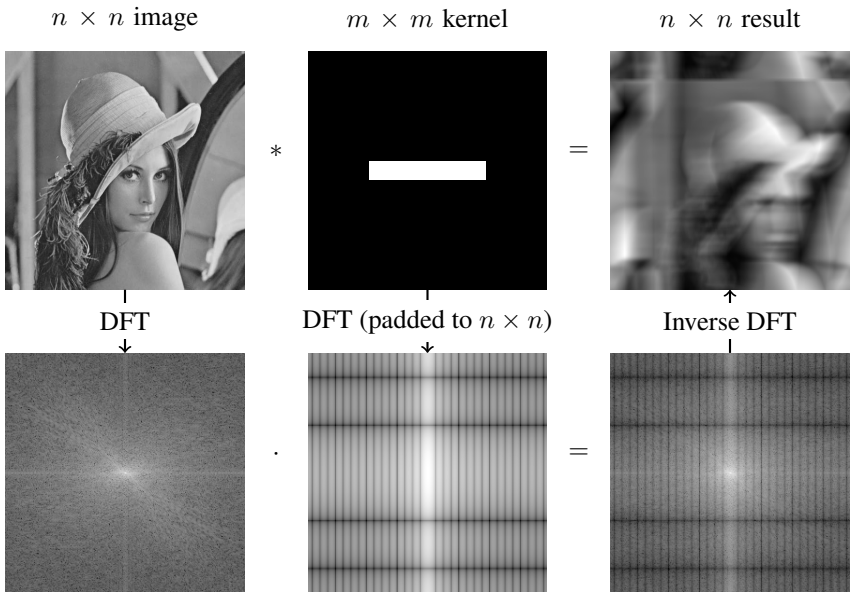

Figure 3.5: DFT convolution without proper padding. The kernel has to be padded to $n \times n$ before the DFT in order to obtain two signals of the same size for multiplication.

paadded to be of the same size of the image, which yields an aliased image. For images the act of convolving an image u with a kernel κ is the same as applying a linear filter to the image. Although the resulting convolution is of size $n + m - 1 \times n + m - 1$ we are mostly interested in the convolution in the $n \times n$ area corresponding to the original image. For this reason we crop the convolution result. Figure 3.6 shows a convolution procedure with ”correct” padding, avoiding aliasing.

One problem with zero-padding is that the edges of the resulting convolved image will be darkened. This is unrealistic or unwanted for mathematical models and linear filters. This effect is worse for kernels κ with larger resolution. The convolution around the edges of the image requires knowledge about the intensities outside the image, which is information we generally do not have. One way to lessen the impact of this effect is so called ”egde”-padding, where instead of padding the image with zeros, we pad it with the value at the edge pixel of the image. This procedure is shown in 3.7. When doing this one must be careful with shifting and cropping the image. Note that when applying edge padding, the ”edge” part of the resulting convolution will be aliased. To illustrate this, figure 3.8 shows the periodic extension of the three resulting convolutions of the proposed convolution methods.

3.1.6 The Fast Fourier Transform

The convolution theorem is a crucial result, as it allows us to calculate the convolution between two signals by the DFT. The naive calculation of the 2D convolution can be done in $\mathcal{O}(n^2m^2)$ operations.

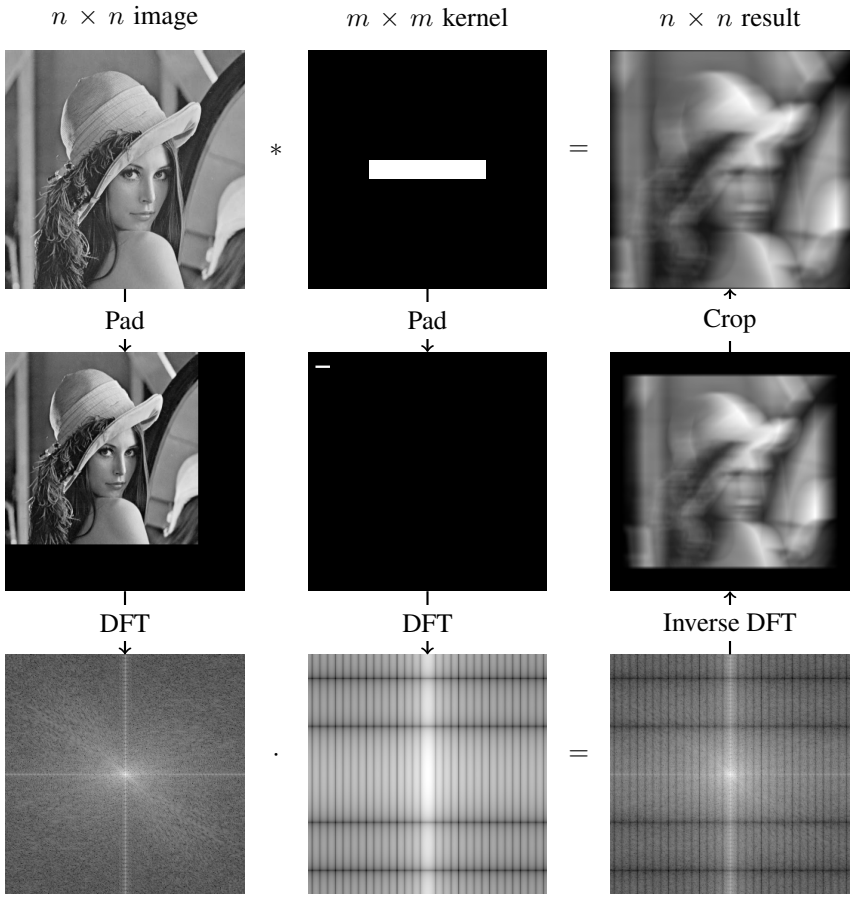

Figure 3.6: DFT convolution with proper zero-padding. Both the image and kernel are padded to a resolution of $(n + m - 1) \times (n + m - 1)$.

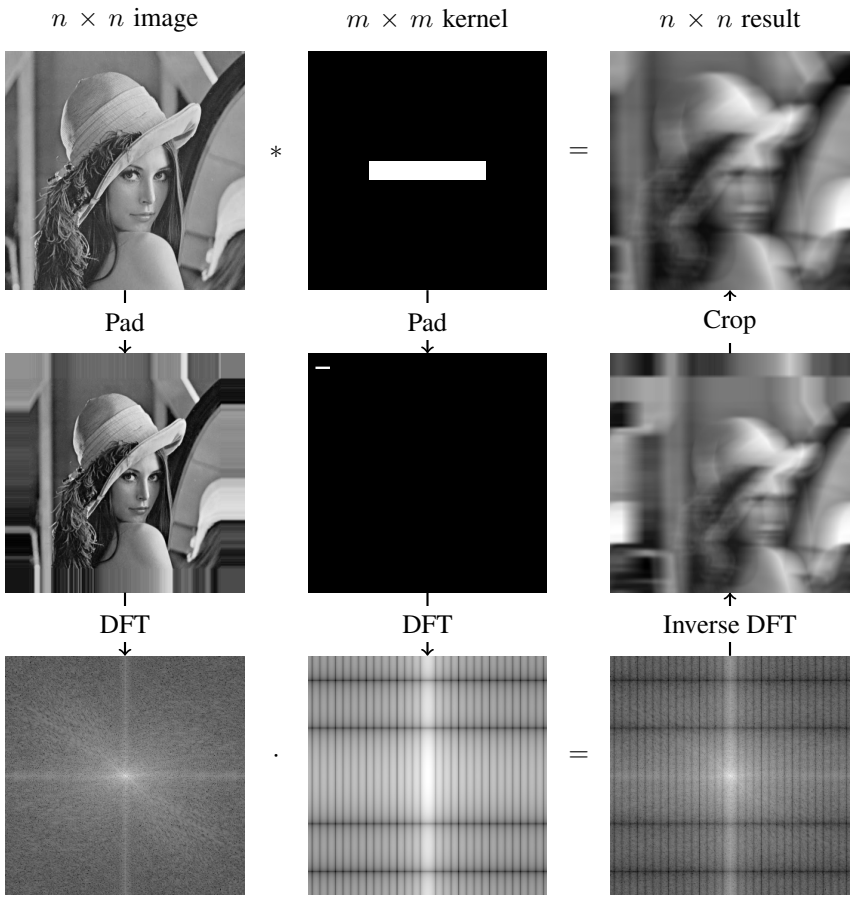

Figure 3.7: FFT convolution with "edge" image padding. Both the image and kernel are padded to a resolution of $(n + m - 1) \times (n + m - 1)$. Note that after the inverse DFT the image is in the lower right corner, not in the middle as in figure 3.6.

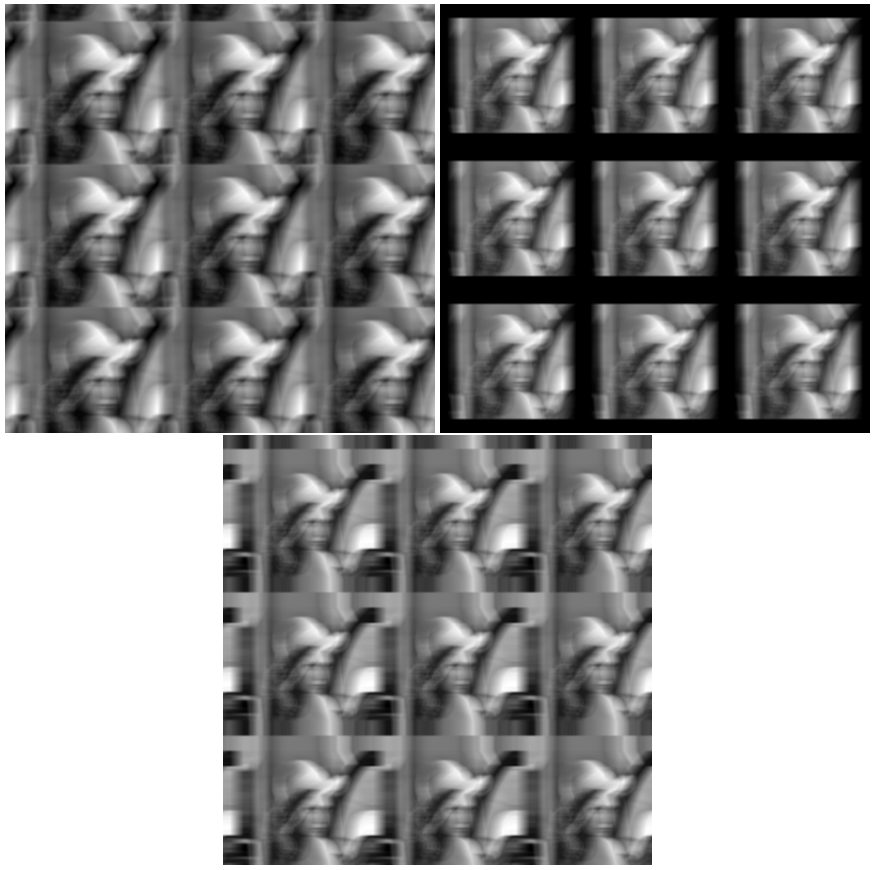

Figure 3.8: Periodic extenstions of the resulting convolutions in figure 3.5, 3.6 and 3.7 respectively.

The DFT can be calculated using the **Fast Fourier Transform** (FFT) algorithm, which has many efficient implementations.

The complexity of the FFT convolution becomes $\mathcal{O}(n^2 \log n)$, a huge improvement in many cases. Because we are working with real signals, half of the frequencies are redundant, so we can use an FFT algorithm specialized for this case. It is also possible to use different trigonometric transforms, like the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST).

It is worth mentioning that there exists other fast convolution algorithms. One of them being consecutive convolutions with separable kernels, which means that

$$\kappa = \kappa_x * \kappa_y, \tag{3.92}$$

for two kernels $\kappa_x \in \mathbb{R}^{m \times 1}$ and $\kappa_y \in \mathbb{R}^{1 \times m}$. In this case the convolution can be calculated as $(u * \kappa_x) * \kappa_y$ in $\mathcal{O}(n^2 m)$ when done naively. Another algorithm is the so-called overlap-and-add algorithm, which is well-suited when the resolution of the signal u and the kernel κ is of very differing size.

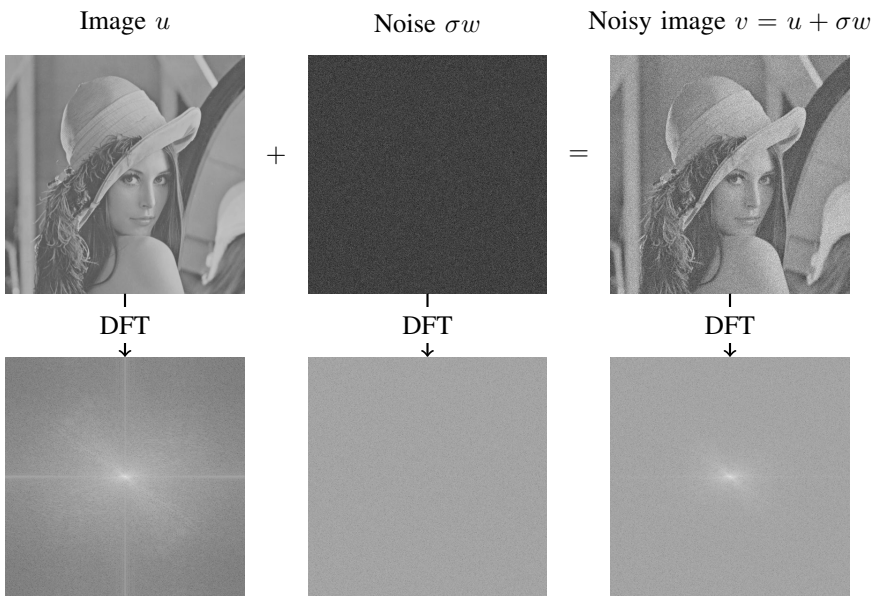

Figure 3.9: The power spectrum of noise added to an image.

3.2 Fourier analysis and inverse problems

We can use the tools of Fourier analysis to better understand deconvolution and denoising.

If we first investigate the denoising model

$$v = u + \sigma w, \tag{3.93}$$

where w is a random vector with uncorrelated, zero mean and finite, equal variance elements. In this case, w is usually called **white noise**, and has the property that the power spectrum is constant. For finite signals, the power spectrum slightly deviates from a constant spectrum because of the random properties of the noise. In the case where w is Gaussian distributed, we usually call this Gaussian white noise.

Figure 3.9 shows what happens to the spectrum of an image when noise is added. From the figure, we see that the DFT of the noise vector is a nearly constant spectrum, while the DFT of an image usually has most of its weight clustered around the center, corresponding to low frequencies.

In this text we have mentioned the word **filter** without properly defining it. A filter is a process that removes certain components of a signal, usually certain frequencies. In particular, a **band-pass filter**. is a filter which "passes" certain frequencies of a signal, and removes others. A particular case of a band-pass filter is a **low-pass filter** which preserves all low frequencies up to a maximal frequency. A linear filter usually corresponds to a multiplication in Fourier space (removing and amplifying certain frequencies) which corresponds to a convolution in signal space. Examples of linear filters are Gaussian filters and QV regularization. An example of a non-linear filter is TV regularization. All of these

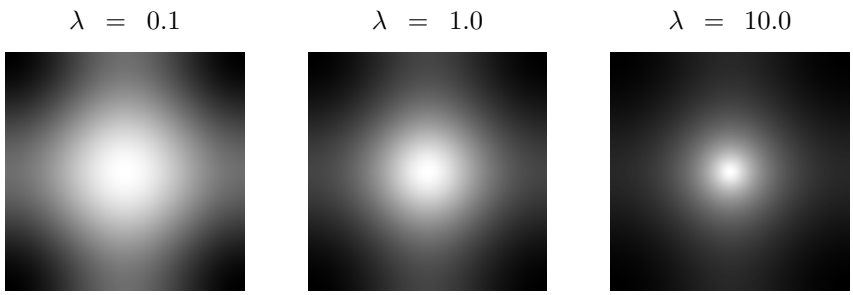

Figure 3.10: DFT of QV solution operator for different λ.

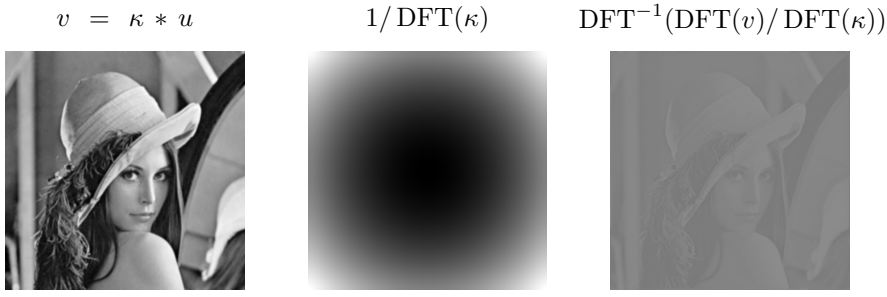

Figure 3.11: Example of deconvolution of Gaussian blur. The Gaussian blur has a kernel of size 15×15 and $\sigma = 1.2$. The condition number of the resulting operator is $\mathcal{K}(A) = 3.7 \times 10^5$. The resulting deconvolution has $\mathrm{PSNR} = 16.43$

filters are low-pass filters. Low-pass filters usually smooth a signal by not allowing large frequencies, corresponding to "fast" variations. This smoothing corresponds to noise reduction when the original signal consists mostly of lower frequencies, like images. Figure 3.10 shows the spectrum of the solution operator $S_{\mathrm{QV}}(\lambda)$ for denoising for different λ. We see that the spectrum is large near the center, corresponding to low frequencies, and small near the edges.

Figure 3.9 shows that we can remove large frequencies, as they mostly correspond to noise, not the signal. We can therefore apply a low-pass filter like Gaussian blur, QV regularization and TV regularization to remove noise.

3.2.1 Deconvolution

We will now investigate some of the properties of deconvolution. We interpret deconvolution as removing a linear convolution with edge padding as was shown in figure 3.7. We remember that a deconvolution is also a convolution, so we apply the same edge padding method to deconvolution. An example of this "naive" deconvolution is shown in figure 3.11.

While Gaussian blur is a low-pass filter, we see that the inverse is a high-pass filter. While the blur operator A is not too ill-conditioned, the main problem with the decon-

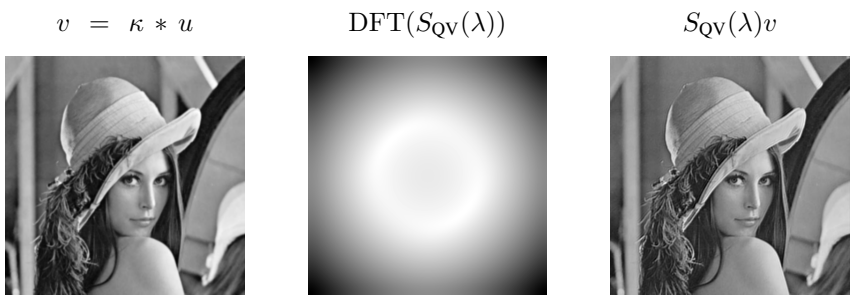

Figure 3.12: Example of deconvolution of Gaussian blur using QV regularization and $\lambda = 0.05$. The Gaussian blur is the same as in figure 3.10. The resulting deconvolution has $\mathrm{PSNR} = 33.66$.

volution is the ringing effects that occur. The ringing occurs because of the sharp jump caused by the periodic extension of the original image. This ringing is especially visible around the edges of the image. This is also for a problem without noise. If noise is added, the resulting deconvolution will amplify that noise, especially if the blur is more ill-conditioned.

The ringing and noise can be alleviated with regularization. An example of this is shown in figure 3.12. The resulting deconvolution using QV regulariztion is much better than the "naive" deconvolution. Note that the solution operator is a low-pass filter. The QV deconvolution does however have some ringing near the edges, as well as some extra QV blur. This can be improved (or worsened) for different parameter choices for $S_{\mathrm{QV}}(\lambda)$. As with the denoising problem, choosing the correct parameter is of great importance. We will later see that the results for TV regularization will be similar as for QV regularization.

Chapter 4

Image resampling

Image resampling amounts to transforming one discrete image into a similar discrete image with different resolution. Ideally, this should be done without losing much information about the image. Image resampling can be done to create an image with smaller resolution, called **downsampling**, and to create images of larger resolution, called **upsampling**. In addition, these methods can be used to resample an image with the same resolution. Aliasing becomes a problem when performing image resampling, and we will also discuss techniques to avoid aliasing.

Image resampling is traditionally done using interpolation, which can often be described as linear operators, usually convolutions. The idea of interpolation is to find a continuous signal interpolated from a discrete signal, then sample from this new continuous signal.

Moreover, it is possible to describe image resampling as an inverse problem

$$v = Au + \sigma w \tag{4.1}$$

where v is a resampled image of u and A is either a downsampling or upsampling operator. The goal of resampling is then to "undo" a downsampling or upsampling. This approach is mainly used for upsampling, as downsampling can usually be done with traditional interpolation techniques. Newer methods of upsampling also merge into the concept of **super-resolution** imaging. The goal of super-resolution imaging is often to extract information from several similar images to produce an image of better quality and higher resolution.

We wish to resample an $n \times n$ image u to a $n' \times n'$ image u'. We define the resampling ratio α

$$\alpha = \frac{n'}{n} \tag{4.2}$$

In the downsampling problem, we have $\alpha < 1$, while in the upsampling problem we have $\alpha > 1$, usually an integer value. We define a pixel coordinate of the grid of the original image as (x, y), and the pixel coordinate of the new grid as (x', y'). The original grid

consists of all pairs of integers satisfying

$$\{x \in \mathbb{N} | x < n\} \times \{y \in \mathbb{N} | y < n\}. \tag{4.3}$$

It is not enough to just have the resampling ratio α, we also need to choose a new grid and a method for calculating the value of (x', y'). We need to transform between the grid of the new image and the old image. This transform is obviously not unique, and there are infinitely many ways to do this transformation. We can think of this as choosing a continuous coordinate transform B satisfying

$$B(x', y') = (x, y). \tag{4.4}$$

An illustration is shown in figure 4.1.

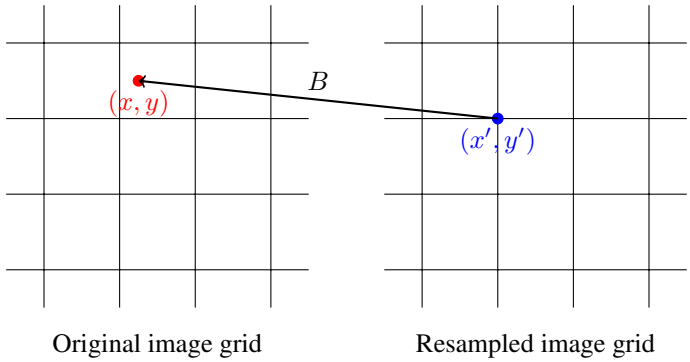

Figure 4.1: The continuous coordinate transform B transforms a pixel of the resampled image grid (x', y') to a pixel of the original image (x, y). Note that the point (x, y) does not necessarily hit a point on the grid of the original image.

We seldom need to mathematically describe the operator B, but it is a useful abstraction. For simplicity, we usually choose the resampled grid to have equidistant, possibly shifted points, which means that the coordinate transform B is a particular class of affine operators.

The point (x', y') need not correspond with a coordinate (x, y) on the grid of the original image, i.e a coordinate where we know the image value. An illustration of two different choices for the new grid is shown in figure 4.2. A resampling algorithm then requires us to choose a method for approximating such pixels, which can be done through interpolation.

There exists many different linear interpolation methods, or more specifically, interpolation kernels. Many interpolation methods can be written as separable convolutions. In this text we will discuss nearest-neighbour interpolation, bilinear interpolation, bicubic interpolation and Lanczos interpolation. These methods can be interpreted as a convolution with between a discrete signal, interpreted as a Dirac comb, and a continuous kernel w

$$u'(x) = \sum_{y=-\infty}^{\infty} w(x - y)u(x). \tag{4.5}$$

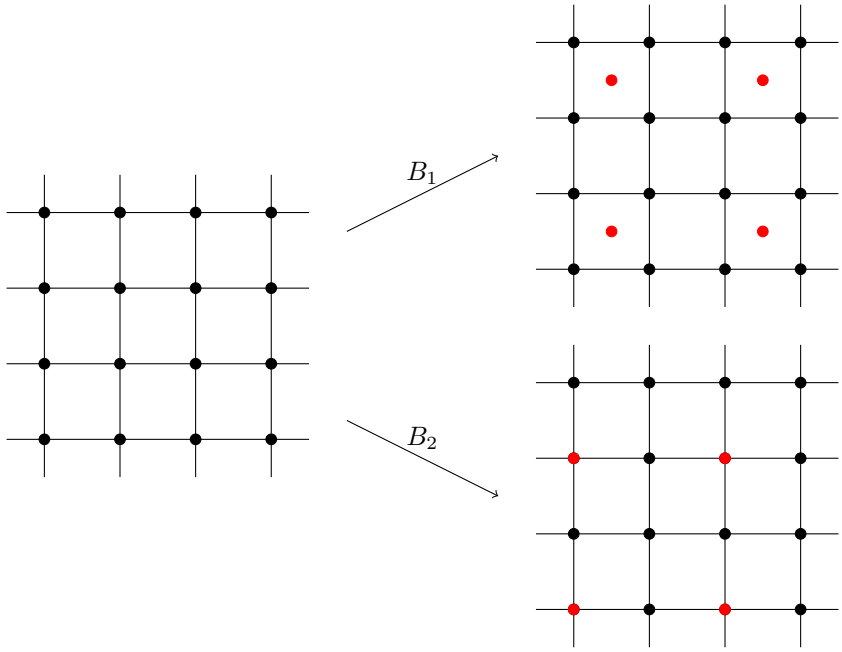

Figure 4.2: Illustration of two different coordinate transforms B_1 and B_2 for downsampling with $\alpha = 1/2$. B_1 maps new points (x', y') between the original points (x, y) while B_2 maps new points onto a subset of the old points.

The act of downsampling is then to sample a discrete signal from the interpolated function u'.

The simplest interpolation method is **nearest-neighbour interpolation**. This method simply consists of choosing the interpolated pixel to be the closest pixel in the original image. The pixel (x', y') of the new image u' satisfies

$$u'(x', y') = u(\lfloor x' + 0.5 \rfloor, \lfloor y' + 0.5 \rfloor). \tag{4.6}$$

This is a very simple and easy to implement method. It can also be expressed as a convolution with a square windowing kernel in 1D

$$\kappa_{1Dnn}(x) = \begin{cases} 1, & -0.5 \leq x \leq 0.5 \\ 0, & \text{otherwise} \end{cases} . \tag{4.7}$$

The nearest neighbour kernel easily extends to 2D

$$\kappa_{2Dnn}(x, y) = \begin{cases} 1, & -0.5 \leq x \leq 0.5, -0.5 \leq y \leq 0.5 \\ 0, & \text{otherwise} \end{cases} . \tag{4.8}$$

A more complex method is **bilinear interpolation**. In 1D, linear interpolation amounts to a weighted sum between the two closest points. The point we want to interpolate, x'

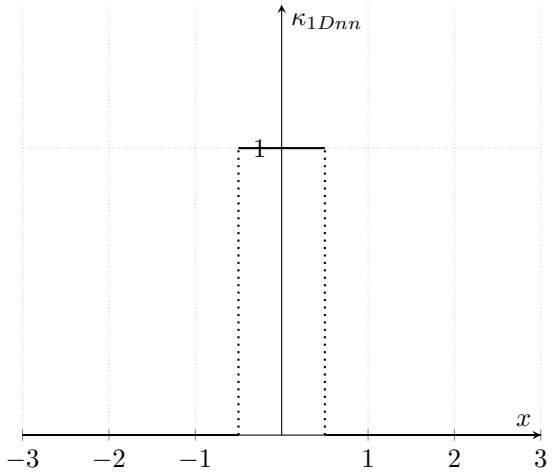

Figure 4.3: The kernel for nearest neighbour interpolation

has two closest points on the original grid x_1 and x_2 satisfying $\lfloor x' \rfloor = x_1$ and $\lceil x' \rceil = x_2$. Given our scaling, the points also satisfy $x_2 = x_1 + 1$. Linear interpolation can then be written as

$$u'(x') = (x_2 - x') \cdot u(x_1) + (x' - x_1)u(x_2). \tag{4.9}$$

Note that the equation satisfies $u'(x_1) = u(x_1)$ and $u'(x_2) = u(x_2)$, meaning the method reduces to nearest neighbour interpolation when a new pixel aligns with a pixel on the existing grid. Linear interpolation can also be interpreted as the line between the two points, or a weighted mean of the two points. It is easy to express as a convolution with a kernel

$$\kappa_{1Dlin}(x) = \begin{cases} 1 - |x|, & |x| < 1 \\ 0, & \text{otherwise} \end{cases}. \tag{4.10}$$

In 2D, bilinear interpolation corresponds to a linear interpolation between two linear interpolations. In other words, it depends only on the closest four pixels, and is a weighted mean of these pixels. One begins with linear interpolation between two pairs of pixels, and then interpolate between the appropriate points on these two lines. Writing this out is rather technical, and it is more easily expressed as a convolution with a kernel

$$\kappa_{2Dlin}(x, y) = \kappa_{1Dlin}(x) \cdot \kappa_{1Dlin}(y) = \begin{cases} 1 - |x| - |y| + |xy|, & |x| < 1, |y| < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{4.11}$$

Both nearest-neighbour and bilinear interpolation are simple naive methods that do not take into account the spectrum of the signal u. We seek ways of interpolating a band-limited signal

A band-limited, continuous, real signal has a spectrum ranging from $-\xi_{\max}$ to $\xi_{\max}$, while a sampled discrete version has the same spectrum but repeated infinitely. To reconstruct the signal, we should try to isolate this original spectrum. This can be done by

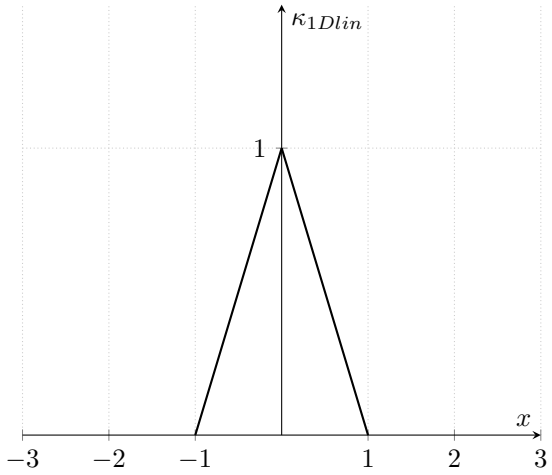

Figure 4.4: The kernel for linear interpolation.

multiplying the spectrum $U(\xi)$ with a square windowing function

$$U(\xi) \cdot H_\pi(\xi) = U(\xi) \cdot \begin{cases} 1, & -\pi \leq \xi \leq \xi_{\max} \\ 0, & \text{otherwise} \end{cases} . \tag{4.12}$$

This is an ideal low-pass filter, as it perfectly passes low frequencies while removing frequencies beyond ξ_s As we have repeated several times, this multiplication in Fourier space corresponds to a convolution in signal space. The inverse Fourier transform of the square windowing function $H(\omega)$ is the so-called Sinc function

$$\mathcal{F}^{-1}(H_\pi(\xi)) = \mathrm{Sinc}(x) = \begin{cases} 1, & |x| = 0 \\ \frac{\sin(\pi x)}{\pi x}, & \text{otherwise} \end{cases} . \tag{4.13}$$

For a band-limited signal, the optimal interpolation method is a convolution with the Sinc function

$$\kappa_{1Dsinc}(x) = \mathrm{Sinc}(x). \tag{4.14}$$

Note that the sinc function is 0 for all integer values of x.

In 2D, this corresponds to a convolution with a 2D Sinc function

$$\kappa_{2Dsinc}(x, y) = \kappa_{1Dsinc}(x)\kappa_{1Dsinc}(y) = \frac{\sin(\pi x)}{\pi x} \frac{\sin(\pi y)}{\pi y}. \tag{4.15}$$

This is called **Sinc interpolation**, also called the Whittaker-Shannon interpolation formula. It can be shown that using Sinc interpolation can perfectly recreate a continuous signal from a properly sampled, band-limited, discrete signal. At least ideally, but as we will see there are some problems in practice.

While the nearest-neighbour interpolation used only the closest point, and linear interpolation uses the closest two points, Sinc interpolation uses infinitely many points. Sinc

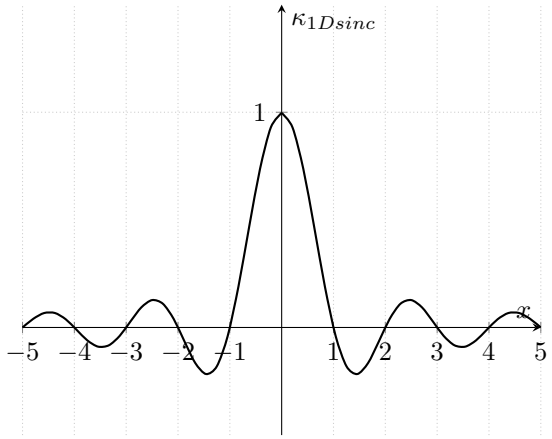

Figure 4.5: The kernel for Sinc interpolation.

interpolation can potentially give negative weight to some pixels. In order to perfectly recreate a continuous signal, we need information of the entire signal for each point, not just the local information. This is obviously problematic as it is makes Sinc interpolation not calculateable for an infinite signal, and potentially costly for a finite one. Most advanced interpolation methods are approximations to the Sinc function.

One of the most common interpolation methods is **Bicubic interpolation**, which approximates the Sinc function with a cubic function. The kernel is given as

$$\kappa_{1Dcub} = \begin{cases} (-a+2)|x|^3 + (a-3)|x|^2 + 1, & 0 \leq |x| < 1 \\ -a|x|^3 + 5a|x|^2 - 8a|x| + 4a, & 1 \leq |x| < 2 \\ 0, & \text{otherwise} \end{cases} \tag{4.16}$$

, where a is a tuning parameter usually set to 1.

The bicubic interpolation method can also weigh pixels negatively, but only uses information from the neighbouring four points.

For downsampling, it is most common to choose the resampling ratio such that the pixels of the downsampled image aligns with the pixels of the original image. In this case all methods is equivalent to nearest neighbour resampling. For example, when $\alpha = 1/2$, meaning we downsample an $n \times n$ image into a $n/2 \times n/2$ image, we can downsample by choosing every other pixel of the original image. This will obviously lead to aliasing, as we are only using one fourth of the pixels of the original image to calculate the downsampled image.

To avoid aliasing, one usually applies an **anti-aliasing filter**. An anti-aliasing filter is essentially the same as a denoising filter, in that it attempts to remove aliased frequencies. For images, these are usually high frequencies, which means that anti-aliasing filters corresponds to a low-pass filter, like a Gaussian blur. An anti-aliasing smooths the pixels intensities, so that information from neighbouring pixels are added to downsampled pixels. The anti-aliasing filter can be applied both before and after the downsampling procedure. We will investigate this more in a later chapter.

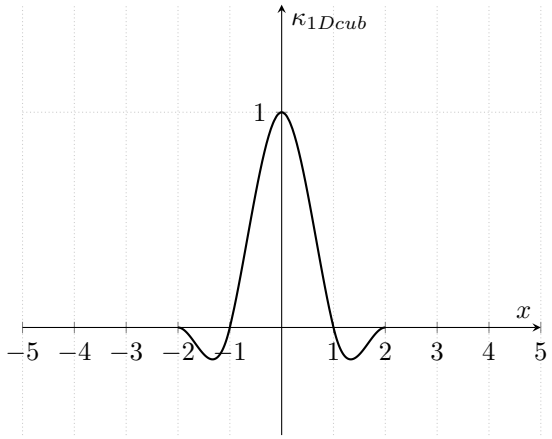

Figure 4.6: The kernel for cubic interpolation with the parameter $a = 1$. Note the similarity with the Sinc function.

For upsampling, it is also possible that some of the pixels of the upsampled image corresponds to the original image, but there will also be new pixels that lie in between pixels of the original image. Some of the different upsampling methods is shown in figure 4.7.

Nearest-neighbour interpolation usually yields ”blocky” or ”pixelated” images. This is sometimes a wanted effect, but is often avoided. The more advanced methods yields smoother results. In the case of linear interpolation, the upsampled version can in some cases be too smooth, creating blur. For bicubic interpolation, and any interpolation method that can weigh pixels negatively, an unwanted effect that is usually called **ringing** can occur. Ringing occurs near sharp transitions of a band-limited signal. It is equivalent to the Gibbs phenomenon, which refers to oscillations in the Fourier series near discontinuities of a periodic function. We saw the ringing phenomenon when we discussed deconvolution in chapter 3. Ringing can also be interpreted as the oscillations in the Sinc function. More intuitively, a sharp transition in a signal requires high frequencies. When these high frequencies are removed, the sharp transitions will be inaccurate, and will have to be created with smaller frequencies.

Figure 4.7: Left is a 56×56 image. All other images are upsampled images with $\alpha = 2.0$. The upper middle image is created using nearest-neighbour interpolation. The upper right image is created using bilinear interpolation. The bottom image is created using bicubic interpolation.

Chapter 5

Convex Analysis and Numerical Methods

This chapter will provide required definitions and results from the field of convex analysis and optimization. The aim is for this chapter to be self-contained, and in this pursuit we will omit some proofs and assume standard optimization results are known. The chapter is loosely based on chapter 3, 13, 15, 16 and 28 in the rather extensive work of Bauschke et al. [3]

The field of convex optimization concerns optimization of convex functions on convex sets. While mathematical optimization is in general an NP-hard problem , many convex optimization problems allow for more efficient solution algorithms. The main appealing property of convex optimization problems is that local solutions are also global solutions, and for differentiable functions every stationary point is a global solution. [17] The class of convex optimization problems also contain many common problems, such as linear programming and least square problems.

As discussed earlier least square solutions are simple solutions (but not necessarily good ones!) to inverse problems. Many optimization problems that arise in inverse problems are convex, but not differentiable. In such cases, "traditional" line search and trust region methods fail. The goal is then to have a mathematical framework and numerical methods for solving convex problems, which is why we turn to convex analysis.

5.1 Convex analysis

We begin by assuming U to be a Hilbert space, but not all definitions and proofs require this structure.

Definition 12. *(Convexity) A function $f : U \to \mathbb{R}$ is convex if for all $u, v \in U$ and $0 \leq t \leq 1$ we have*

$$f(tu + (1-t)v) \leq tf(u) + (1-t)f(v). \tag{5.1}$$

A subset C of U is convex if for all $u, v \in U$ and $0 \leq t \leq 1$ we have

$$tu + (1-t)v \in C. \tag{5.2}$$

In other words, a set is convex if line segments between points of the set is contained in the set. A function is convex if line segments between points on the graph of the function lies on or above the graph of the function.

Definition 13. *(Projection) Let C be a closed, convex subset of U. Then the projection $\pi_C(u)$ is the solution of the problem*

$$\pi_C(u) = \min_{v \in C} \|v - u\|^2.$$

Note that the mapping $v \mapsto \|v-u\|^2$ is strictly convex and coercive, meaning the projection exists and is unique. For convex subsetss C, there exists an useful way of characterising projections, which will be stated without proof.

Theorem 4. *Let $\pi_C(u)$ be the projection onto the closed, convex subset C. Then*

$$v = \pi_C(u) \iff v \in C \text{ and } \langle u - v, w - v \rangle \leq 0, \forall w \in C.$$

Example 1. *Let C be the closed ball around 0 with radius $\lambda > 0$*

$$C = B_\lambda = \{u \in U : \|u\| \leq \lambda\}.$$

The projection onto this subset satisfies

$$\pi_{B_\lambda}(u) = \frac{\lambda u}{\max\{\lambda, \|u\|\}}$$

We can use Theorem 4 to characterize a projection onto this subset. Let $v = \frac{\lambda u}{\max\{\lambda, \|u\|\}}$). For $\|u\| \leq \lambda$, we have that $v = u$, and thus the variational inequality holds trivially. For $\|u\| \geq \lambda$ we have that $v = \frac{\lambda u}{\|u\|}$, and the variational inequality yields

$$\begin{aligned}
\langle u - \frac{\lambda u}{\|u\|}, w - \frac{\lambda u}{\|u\|} \rangle &= \langle u, w \rangle - \lambda \|u\| - \frac{\lambda}{\|u\|} \langle u, w \rangle + \lambda^2 \\
&= -\|u\| \lambda \left(1 - \frac{\lambda}{\|u\|}\right) + \langle u, w \rangle \left(1 - \frac{\lambda}{\|u\|}\right) \\
&= -\|u\| \lambda \left(1 - \frac{\lambda}{\|u\|}\right) + \|u\| \lambda \left(1 - \frac{\lambda}{\|u\|}\right) = 0.
\end{aligned}$$

Thus $\pi_{B_\lambda}(u)$ is the projection onto B_λ.

To exclude uninteresting functions we first begin by defining a class of convex functions.

Definition 14. *(Domain) The domain of a function $f : U \to \mathbb{R} \cup \{+\infty\}$ is*

$$\mathrm{dom}(f) = \{u \in U : f(u) < +\infty\}$$

Definition 15. *(Proper convex function) A proper convex function is a convex function $f : U \to \mathbb{R} \cup \{+\infty\}$ so that $\mathrm{dom}(f)$ is non-empty, i.e $\exists u \in U$ s.t $f(u) < +\infty$.*

In particular, we denote

$$\Gamma_0(U) = \{f : U \to \mathbb{R} \cup \{+\infty\} \text{ proper, convex and lower semi-continuous}\} \tag{5.3}$$

Example 2. *When f is allowed to take on $+\infty$, convex functions are no longer guaranteed to be lower semi-continuous. For example, an important function is the characteristic of a convex set C,*

$$i_C(u) = \begin{cases} 0, & u \in C \\ +\infty, & u \notin C \end{cases}.$$

Now choose $C = (0, +\infty]$. The characteristic function in this case is convex. The inequality

$$i_C(tu + (1-t)v) \leq t i_C(u) + (1-t) i_C(v), 0 \leq t \leq 1$$

always holds because if there is a $+\infty$ on the left side, either $i_C(u) = +\infty$ or $i_C(v) = +\infty$. However, it is not lower semi-continuous, as $\liminf_{u \to 0^+} i_C(u) = 0$, but $i_C(0) = +\infty$.

The characteristic function is lower semi-continuous for closed C. Recall that unconstrained optimization problems have solutions if the function we are minimizing is coercive and lower semi-continuous.

5.1.1 Convex conjugates and their properties

We can now begin to introduce some of the main tools in convex analysis.

Definition 16. *(Convex Conjugate) For a function $f : U \to \mathbb{R} \cup \{+\infty\}$, we define its convex conjugate $f^* : U \to \mathbb{R} \cup \{+\infty\}$ (also called Legendre-Fenchel transform)*

$$f^*(p) = \sup_{u \in U} [\langle u, p \rangle - f(u)]. \tag{5.4}$$

The power of the convex conjugate will become evident when discussing optimality conditions for convex optimization problems. For now, let us try to calculate the conjugate of a few functions.

Example 3. *The characteristic function mentioned earlier is interesting because it turns unconstrained problems into constrained ones, for example*

$$\arg\min_{u \in U} [f(u) + i_C(u)] = \arg\min_{u \in C} f(u), \tag{5.5}$$

in the sense that the two problems are identical. Assume C is a ball with radius λ:

$$C = B_\lambda = \{u \in U : \|u\| \leq \lambda\}$$

. We can calculate the convex conjugate of the indicator function on this set

$$f^*(p) = \sup_{u \in U}[\langle u, p\rangle - f(u)] = \sup_{\|u\| \leq \lambda} \langle u, p\rangle = \lambda \|p\| \tag{5.6}$$

Example 4. *Another useful calculation is the adjoint of $f(u) = \frac{1}{2}\|Au - v\|^2$ for a linear operator $A : U \to V$, $u \in U$, $v \in V$.*

$$f^*(p) = \sup_{u \in U}[\langle u, p\rangle - \frac{1}{2}\|Au - v\|^2] = \sup_{w \in V, w = Au} [\langle A^{-1}w, p\rangle - \frac{1}{2}\|w - v\|^2].$$

The problem is differentiable, so the supremum satisfies,

$$A^{-*}p = w - v.$$

Inserting this we get,

$$\begin{aligned} f^*(p) &= \langle A^{-*}p + v, A^{-*}p\rangle - \frac{1}{2}\|A^{-*}p\|^2 \\ &= \|A^{-*}p\|^2 + \frac{1}{2}\|v\|^2 - \frac{1}{2}\|A^{-*}p - v\|^2 \\ &= \frac{1}{2}\|A^{-*}p + v\|^2 - \frac{1}{2}\|v\|^2. \end{aligned}$$

A tool that is frequently used Young's inequality.

Lemma 7. *(Young's Inequality) For all $u, v \in U$ we have*

$$\langle u, v\rangle \leq f(u) + f^*(v) \tag{5.7}$$

Proof. From the definition of convex conjugate we have

$$f^*(v) \geq \langle u, v\rangle - f(u),$$

and the result trivially follows. □

Theorem 5. *Let $f \in \Gamma_0(U)$, then $f^* \in \Gamma_0(U)$.*

Theorem 6. *Let $f : U \to \mathbb{R} \cup \{+\infty\}$ be proper. Then $f^{**} \leq f$ with equality if and only if $f \in \Gamma_0(U)$.*

Proof. Apply Young's inequality

$$f(u) \geq \langle u, v\rangle - f^*(v), \forall u, v \in U.$$

Because the inequality holds for all $u, v \in U$, it also holds if we apply a supremum on one side,

$$f(u) \geq \sup_{v \in U}[\langle u, v\rangle - f^*(v)] = f^{**}(u).$$

□

Example 5. *Continuing from Example 3, we can put Theorem 5 and 6 to the test.*
Let us try to find the conjugate of $f^*(v) = \lambda\|v\|$

$$f^{**}(u) = \sup_{p\in U^*}[\langle p, u\rangle - f^*(p)] = \sup_{p\in U^*}[\langle p, u\rangle - \lambda\|v\|]. \tag{5.8}$$

If $\|u\| > \lambda$ *the mapping* $v \mapsto \langle p, u\rangle - \lambda\|p\|$ *is unbounded above. If* $\|u\| \leq \lambda$*, the mapping attains a maximum for* $p = 0$*. This means that*

$$f^{**}(u) = \begin{cases} 0, & \|u\| \leq \lambda \\ +\infty, & \|u\| > \lambda \end{cases} = i_{B_\lambda}(u) = f(u)$$

As expected, we have $f = f^{**}$ *and* $f \in \Gamma_0(U)$*,* $f^* \in \Gamma_0(U^*)$*.*

5.1.2 Subdifferentials and their properties

We need a way to define properties of non-differentiable functions. We also need a way of characterizing optimality conditions for non-differentiable functions. Subdifferentials are extensions of differentials of continuous functions.

Definition 17. *(Subdifferential) Let* $f \in \Gamma_0(U)$*. The convex subdifferential* $\partial f(u)$ *of* f *at* u *is*

$$\partial f(u) = \begin{cases} \emptyset, & \text{if } u \notin \operatorname{dom}(f) \\ \{\xi \in U : f(v) \geq f(u) + \langle \xi, v - u\rangle, \forall v \in U\}, & \text{if } u \in \operatorname{dom}(f) \end{cases} \tag{5.9}$$

The elements of $\partial f(u)$ *are called subgradients of* f *at* u*.*

Example 6. *Let* $f(u) = \|u\|$*, which is differentiable everywhere except* $u = 0$*. The subdifferential of* $f(u)$ *at* $u = 0$ *is*

$$\begin{aligned} \partial f(0) &= \{\xi \in U : \|v\| \geq \langle \xi, v\rangle, \forall v \in U\} \\ &= \{\xi \in U : \|\xi\| \leq 1\}. \end{aligned}$$

A line with slope equal to the gradient of a differentiable, convex function always lies under the graph of the function. For non-differentiable, convex functions the subdifferential contains the slopes of all lines that lie under the graph of the function.

Why define subdifferentials this way? One reason is that for differentiable functions, the only element of the sub-differential is the actual gradient of the fucntion.

Lemma 8. *Let* $f : U \to \mathbb{R} \cup \{+\infty\}$ *be convex, lower semi-continuous,* $u \in \operatorname{int}(\operatorname{dom} f)$ *and* f *(Gâteaux) differentiable at* u*, then*

$$\partial f(u) = \{\nabla f(u)\} \tag{5.10}$$

Another reason for defining subdifferentials is that they provide familiar optimality conditions.

Lemma 9. *Let $\hat{u} \in U$ be a minimizer of a convex function $f : U \to \mathbb{R} \cup \{+\infty\}$. Then*

$$0 \in \partial f(\hat{u}). \tag{5.11}$$

Proof. Assume

$$0 \in \partial f(\hat{u}).$$

By the definition of subdifferentials we have

$$\implies f(u) \geq f(\hat{u}), \forall u \in U,$$

meaning $\hat{u}$ is a global minimizer. □

Note that the previous lemma contain the familiar first order necessary optimality condition, namely that optimal points are also stationary points.

The next theorem contains some useful properties of subdiffererntials.

Theorem 7. *(Properties of subdifferentials) Let $f \in \Gamma_0(U)$, $g \in \Gamma_0(P)$ and let $D : P \to U$ be a linear operator. Then:*

1. $$\xi \in \partial f(u) \iff \langle \xi, u \rangle = f(u) + f^*(\xi)$$

2. $$\xi \in \partial f(u) \iff u \in \partial f^*(\xi)$$

3. *Assume $\exists \tilde{u} \in \operatorname{int}(\operatorname{dom} f)$ with $g(\tilde{u}) < \infty$. Then*
$$\partial(f + g)(u) = \partial f(u) + \partial g(u)$$

4. *Define $\partial g(p) = f(Dp)$ and assume $\exists \tilde{p} \in V$ s.t $D\tilde{p} \in \operatorname{int}(\operatorname{dom} f)$. Then*
$$\partial g(p) = D^* \partial f(Dp)$$

Proof. We will omit the proofs of (3.) and (4.) as they are relatively complicated. (1.) Young's inequality yields $\langle \xi, u \rangle \leq f(u) + f^*(\xi)$. The definition of subdifferentials yields

$$f(v) \geq f(u) + \langle \xi, v \rangle - \langle \xi, u \rangle, \forall v \in U,$$

which means that

$$\langle \xi, u \rangle \geq f(u) + \langle \xi, v \rangle - f(v) \geq f(u) + f^*(\xi).$$

This means that $\langle \xi, u \rangle = f(u) + f^*(\xi)$

(2.)

$$\begin{aligned} \xi \in \partial f(u) &\iff \langle \xi, u \rangle = f(u) + f^*(\xi) = f^{**}(u) + f^*(\xi) \\ &\iff u \in \partial f^*(\xi) \end{aligned}$$

□

5.1.3 Duality

As we saw with regularization problems like the total variation problem, we need to be able to solve problems of the form

$$\min_{u\in U}[f(u)+g(Du)].$$

Here $D : U \to P$ is a linear operator. Optimality conditions for such problems can be described by the Fenchel-Rockafellar theorem

Theorem 8. *(Fenchel-Rockafellar) Assume $f \in \Gamma_0(U)$, $g \in \Gamma_0(V)$ and $D \in L(U,V)$. Assume that the primal problem*

$$\min_{u\in U}[f(u)+g(Du)]$$

admits a solution $u^* \in U$ and that there exists $\tilde{u} \in \operatorname{dom} f$ s.t $D\tilde{u} \in \operatorname{int}(\operatorname{dom} g)$. Then

1. *The dual problem*
$$\min_{p\in V}[g^*(p)+f^*(-D^*p)]$$
admits a solution $p^* \in V$.
2. $f(u)+g(Du)+g^*(p)+f^*(-D^*p) \geq 0, \forall u \in U, p \in V$
3. *The following are equivalent:*
   (a) $f(u^*)+g(Du^*)+g^*(p^*)+f^*(-D^*p^*)=0$.
   (b) *u^* solves the primal problem and p^* solves the dual problem.*
   (c) $-D^*p^* \in \partial f(u)$ *and* $p^* \in \partial g(Du)$
   (d) $u^* \in \partial f^*(-D^*p^*)$ *and* $Du^* \in \partial g^*(p^*)$.

Proof. The primal problem has a solution for

$$0 \in \partial(f(u^*)+g(Du^*)) = \partial f(u^*) + D^*\partial g(Du^*).$$

The equality is obtained by applying (3.) and (4.) of theorem 7. We can introduce a new variable p^* to split this into two conditions

$$p^* \in \partial g(Du^*), -D^*p^* \in \partial f(u^*).$$

Applying (2.) of theorem 7, we obatin

$$Du^* \in \partial g^*(p^*), u^* \in \partial f^*(-D^*p^*).$$

We can now rewrite these two conditions into a single condition, and again use (3.) and (4.) of theorem 7

$$0 = Du^* - Du^* \in g^*(p^*) - Df^*(-D^*p^*).$$

Finally, we recognize this as the optimality conditions for a different problem,

$$p^* = \arg\min_{p \in V} g^*(p) + f^*(-D^*p),$$

which we call the dual problem.

We can now use Young's inequality

$$f(u) + g(Du) + g^*(p) + f^*(-D^*p) \geq \langle u, -D^*p \rangle + \langle Du, p \rangle = 0.$$

Equality is attained if and only if $f(u) + f^*(-D^*p) = \langle u, -D^*p \rangle$ and $g(Du) + g^*(p) = \langle Du, p \rangle$, which means 3. (c) and (d) hold, which again mean that 3. (a) and (b) hold. □

We now have optimality conditions for problems on the same form as many regularization problems. The final tool will be a tool that lets us formulate numerical methods for solving the problems, namely proximal operators.

5.1.4 Proximal point operators

Definition 18. *(Proximal point operator) Let $f \in \Gamma_0(U)$. The proximal point mapping $\text{prox}_f : U \to U$ for f is defined as*

$$\text{prox}_f(u) = \arg\min_{\hat{u} \in U} \left[\frac{1}{2} \|\hat{u} - u\|^2 + f(\hat{u}) \right]$$

A useful reformulation is that the proximal operator satisfies

$$0 \in \hat{u} - u + \partial f(\hat{u}) \iff u \in \hat{u} + \partial f(\hat{u}) \iff \text{prox}_f(u) = (I + \partial f)^{-1}(\hat{u}) \quad (5.12)$$

Example 7. *The proximal point mapping of the characteristic function on a set C is*

$$\text{prox}_{i_C}(x) = \arg\min_{\hat{x} \in X} [\frac{1}{2} \|\hat{x} - x\|^2 + i_C(\hat{x})] = \arg\min_{\hat{x} \in C} \frac{1}{2} \|\hat{x} - x\|^2 = \pi_C(x) \quad (5.13)$$

In other words, the proximal operator of the characteristic function onto the set C is the the same as a projection onto the set C.

Lemma 10. *Let $f \in \Gamma_0(U)$ and $u, \xi \in U$. Then for all $\gamma > 0$*

$$\xi \in \partial f(u) \iff u = \text{prox}_{\gamma f}(u + \gamma \xi)$$

Proof.

$$\xi \in \partial f(u) \iff u + \gamma \xi \in u + \gamma \partial f(u) \iff u = \text{prox}_{\gamma f}(u + \gamma \xi)$$

by equation (5.12). □

In other words, proximal operators describe fixed points. An optimal point satisfies

$$0 \in \partial f(u) \iff u = \text{prox}_{\gamma f}(u).$$

We now have all pieces of the puzzle, and are ready to discuss some numerical methods.

5.2 Numerical Methods

We are only interested in optimization problems of the form

$$\min_{u \in U} f(u) + g(Du). \tag{5.14}$$

Most algorithms work for variants of this problem, but we will exclusively apply it to the total variation problem for grayscale images with resolution $n \times n$.

$$\min_{u \in \mathbb{R}^{n^2}} \underbrace{\frac{1}{2}\|Au - v\|^2}_{f(u)} + \underbrace{\lambda \||Du|\|_1}_{g(Du)}. \tag{5.15}$$

The operator D is as defined in equation (2.21). It is worth noting that we have already done much of the analysis needed for the total variation problem in the examples in the last subsection. Furthermore, the optimization problem (5.15) satisfies the conditions of the Fenchel-Rockafellar theorem. Both f and g are continuous and coercive, meaning the problem (5.15) has a solution. Furthermore, both f and g are proper, convex functions, so $f \in \Gamma_0(U)$ and $g \in \Gamma_0(V)$. Finally it is not difficult to choose $\tilde{u} \in \operatorname{dom} f$ so that $D\tilde{u} \in \operatorname{int}(\operatorname{dom} g)$. We have $\operatorname{dom} g = \mathbb{R}^+$, while $\operatorname{dom} g \subset \mathbb{R}^+$, based on A. We can easily choose $\tilde{u}$ so that $g(D\tilde{u})$ takes on any value, including its interior, so this condition also holds.

We have essentially already calculated the conjugates f^* and g^* in examples 3 and 4. The dual problem of (5.15) can be written as

$$\min_{p \in \mathbb{R}^{n^2 \times 2}} \underbrace{\frac{1}{2}\|v - A^{-*}D^*p\|^2 - \frac{1}{2}\|v\|^2}_{f^*(-D^*p)} + \underbrace{i_{B_\lambda}(p)}_{g^*(p)}. \tag{5.16}$$

For the dual variable p we use the notation

$$p_{i,j} = (p_{i,j}^{(x)}, p_{i,j}^{(y)}).$$

The ball in g^* is $B_\lambda = \{p \in V : \||p|\|_\infty \leq \lambda\}$. To be clear, we interpret $|p|$ componentwise (or pointwise), so the ball contains all p so that every component $p_{i,j}$ satisfies $\sqrt{(p_{i,j}^{(x)})^2 +, (p_{i,j}^{(y)})^2} \leq \lambda$.

We can find the adjoint $D^* : \mathbb{R}^{n^2 \times 2} \to \mathbb{R}^{n^2}$ by calculating

$$\begin{aligned}
\langle Du, p \rangle &= \sum_{i<n, j<n} (u_{i+1} - u_{i,j})p_{i,j}^{(x)} + (u_{i,j+1} - u_{i,j})p_{i,j}^{(y)} \\
&= \sum_{1<i<n, 1<j<n} (p_{i-1,j}^{(x)} - p_{i,j}^{(x)} + p_{i,j-1}^{(y)} - p_{i,j}^{(y)})u_{i,j} \\
&- \sum_j p_{1,j}^{(x)} u_{1,j} - \sum_i p_{i,1}^{(y)} u_{i,1} + \sum_j p_{n-1,j}^{(x)} u_{n-1,j} + \sum_i p_{i,m-1}^{(y)} u_{i,n-1} \\
&= \langle u, D^*p \rangle.
\end{aligned}$$

In other words,

$$D^*p = \begin{cases} p^{(x)}_{i-1,j} - p^{(x)}_{i,j}, & 1 < i < n \\ -p^{(x)}_{i,j}, & i = 1 \\ p^{(x)}_{i-1,j}, & i = n \end{cases} + \begin{cases} p^{(y)}_{i,j-1} - p^{(y)}_{i,j}, & 1 < j < m \\ -p^{(y)}_{i,j}, & j = 1 \\ p^{(y)}_{i,j-1}, & j = m \end{cases} \tag{5.17}$$

We will also need the operator norms $\|D\|$ and $\|D^*\|$. In Hilbert spaces, it can be shown that bounded operators D satisfy $\|D\| = \|D^*\|$, so we only need to find $\|D\|$.

The operator norm $\|D\| = \sup\{\|Du\| : u \in \mathbb{R}^{n^2} \text{ with } \|u\| \leq 1\}$ can be found by investigating

$$\begin{aligned} \|Du\| &= \sqrt{\sum_{i<n,j<n} (u_{i+1,j} - u_{i,j})^2 + \sum_{i<n,j<n} (u_{i,j+1} - u_{i,j})^2} \\ &= \sqrt{\sum_{i<n,j<n} u^2_{i+1,j} - 2u_{i,j}u_{i+1,j} + u^2_{i,j+1} - 2u_{i,j}u_{i,j+1} + 2u^2_{i,j}} \\ &\leq \sqrt{\sum_{i<n,j<n} u^2_{i+1,j} + 2|u_{i,j}u_{i+1,j}| + u^2_{i,j+1} + 2|u_{i,j}u_{i,j+1}| + 2u^2_{i,j}} \end{aligned}$$

This upper bound is nearly satisfied when the difference between neighbouring components of u are maximized, i.e when u creates a checkerboard pattern. For $\|u\| = 1$ and ignoring that the sum does not include $i = n$ and $j = n$ this means $u_{i,j} = \pm\frac{1}{n}$. Thus we can approximate $\|D\| \approx \sqrt{8}$ for large n.

Now, onto numerical methods. The main idea of the numerical methods we will formulate is to use the Fenchel-Rockafellar theorem for optimality conditions and use proximal operators to create fixed point methods. When applying such methods, one has to be creative in how to split the functional one wants to minimize. In regularization problems, the splitting is often given by the problem. Additionally, because solutions to the primal and dual problem comes in pairs, setting up the methods to solve either the primal or dual problem can make more sense.

5.2.1 Forward backward splitting

The first method discussed can essentially be derived in two ways, either by using the Fenchel-Rockafellar theorem and attempt to solve the dual problem instead of the primal problem, or by using proximal operators. We will derive it using proximal operators.

Assume g is differentiable, the optimality system is then

$$p = \nabla g(Du), \quad -D^*p \in \partial f(u), \tag{5.18}$$

or

$$p = \nabla g(Du), \quad u = \operatorname{prox}_{\gamma f}(u - \tau D^*p). \tag{5.19}$$

We thus have the iteration

$$\begin{aligned} p_k &\leftarrow \nabla g(Du_k) \\ u_{k+1} &\leftarrow \operatorname{prox}_{\tau f}(u_k - \tau D^*p_k) \end{aligned}$$

This formulation is known as "Forward-Backward splitting". This formulation, and the other proximal operator methods to come, are all relatively simple to implement, requiring relatively few line of code.

We need to assure some conditions to ensure convergence of the method, stated in the following theorem without proof [8].

Theorem 9. *Let g be (Gâteaux) differentiable and ∇g Lipschitz continuous with constant $L > 0$. The sequence u_k generated by forward-backward splitting converges (weakly) to a solution u^* of the primal problem provided that $0 < \tau_{min} \leq \tau \leq \tau_{max} = \frac{2}{\|D\|^2 L}$*

For the total variation problem it is not easy to apply this directly to the primal problem, but things work out very nicely if we apply it to the dual problem for $A = I$. In other words, we look to satisfy the optimality system

$$u = \nabla f^*(-D^* p), \quad Du \in \partial g^*(p), \tag{5.20}$$

which means we obtain the iteration

$$\begin{aligned} u_k &\leftarrow v - D^* p_k \\ p_{k+1} &\leftarrow \operatorname{prox}_{\tau g}(p_k + \tau D u_k) = \frac{\lambda(p_k + \tau D u_k)}{\max\{\lambda, |p_k + \tau D u_k|\}} \end{aligned}$$

Note that $\nabla f^*(-D^* p) = v - D^* p$ is Lipschitz continuous with Lipschitz constant $L = 1$. Furthermore, with $\|D\|^2 \leq 8$, we should choose $\tau \leq \frac{1}{4}$.

5.2.2 Primal-Dual methods

The idea of this algorithm is to do fixed point iteration in both the primal and the dual variable. The methods we state here are based on the works of Chambolle and Pock[7]. We can use the optimality conditions

$$-D^* p \in \partial f(u), \quad Du \in \partial g^*(p), \tag{5.21}$$

or

$$p = \operatorname{prox}_{\sigma g^*}(p + \sigma D u), \quad u = \operatorname{prox}_{\tau f}(u - \tau D^* p). \tag{5.22}$$

to obtain the iteration

$$\begin{aligned} p_{k+1} &\leftarrow \operatorname{prox}_{\sigma g^*}(p_k + \sigma D u_k) \\ u_{k+1} &\leftarrow \operatorname{prox}_{\tau f}(u_k - \tau D^* p_{k+1}). \end{aligned}$$

This is called the Arrow-Hurwicz method.

Theorem 10. *The sequence (u_k, p_k) generated by the Arrow-Hurwicz method converges (weakly) to the primal-dual solution (u^*, p^*) provided that $\sigma\tau < \frac{1}{\|D\|^2}$.*

A problem with this method is that the update for p uses the u from the previous iteration while the update for u uses the p from the current iteration. We can try to better the method by adding an extrapolation step for u, and obtain the iteration

$$\begin{aligned} p_{k+1} &\leftarrow \operatorname{prox}_{\sigma g}(p_k + \sigma D\hat{u}_k) \\ u_{k+1} &\leftarrow \operatorname{prox}_{\tau f}(u_k - \tau D^* p_{k+1}) \\ \hat{u}_{k+1} &\leftarrow u_{k+1} + \theta(u_{k+1} - u_k). \end{aligned}$$

This method is called the Chambolle-Pock method. The parameter $\theta \geq 0$ is usually set to 1.0. If we set $\theta = 0$ we again obtain the Arrow-Hurwicz method.

Theorem 11. *The sequence (u_k, p_k) generated by the Chambolle-Pock method with $\theta = 1$ converges (weakly) to the primal-dual solution (u^*, p^*) provided that $\sigma\tau \leq \frac{1}{\|D\|^2}$.*

For the total variation problem, we should choose $\sigma\tau < \frac{1}{8}$.

Chambolle also proposed an accelerated version in the case where g or f^* is uniformly continuous, which is true for the total variation problem. Here the step lenghts τ and σ are updated for each iteration to hopefully yield better convergence. The iterations become

$$\begin{aligned} p_{k+1} &\leftarrow \operatorname{prox}_{\sigma g^*}(p_k + \sigma D\hat{u}_k) \\ u_{k+1} &\leftarrow \operatorname{prox}_{\tau f}(u_k - \tau D^* p_{k+1}) \\ \theta_k &= 1/\sqrt{1 + 2\gamma\tau_k}, \tau_{k+1} = \theta_k \tau_k, \sigma_{k+1} = \sigma_k/\theta_k \\ \hat{u}_{k+1} &\leftarrow u_{k+1} + \theta(u_{k+1} - u_k). \end{aligned}$$

Applying the original Chambolle-Pock method to the total variation problem yields the iterations:

$$\begin{aligned} p_{k+1} &= \frac{\lambda(p_k + \sigma D\hat{u}_k)}{\max\{\lambda, |p_k + \sigma D\hat{u}_k|\}} \\ u_{k+1} &= (I + \tau A^* A)^{-1}(u_k - \tau D^* p_{k+1} + \tau A^* v) \\ \hat{u}_{k+1} &= u_{k+1} + \theta(u_{k+1} - u_k). \end{aligned}$$

5.2.3 Total variation deblurring

The update for the primal variable is simple for the denoising problem, but requires solving a linear system for different forward operators A. When A is a convolution operator, $Au = \kappa * u$, we can solve the system directly using the DFT

$$u_{k+1} = \mathrm{DFT}^{-1}\left(\frac{\mathrm{DFT}(u_k - \tau D^* p_{k+1}) + \tau\,\mathrm{DFT}(\kappa)^*\,\mathrm{DFT}(v)}{1 + \tau|\,\mathrm{DFT}(\kappa)|^2}\right). \tag{5.23}$$

To make the process more computationally efficient, we note that $\tau A^* v$ does not change, and only needs to be calculated once. We apply edge padding as was shown in chapter 3. We interpret $(I + \tau A^* A)^{-1}$ as a single convolution, and only apply the padding once. Some care must be taken to ensure that the different terms are padded to the correct sizes

Figure 5.1: Original image u, blurred and noisy image $v = Au + \sigma w$ with $\sigma = 0.02$ and TV reconstructions using $\lambda = 0.0001, 0.001, 0.01, 0.1$ respectively. The blur operator A is a Gaussian blur with a 15×15 kernel and $\sigma_{\text{blur}} = 1.5$.

and cropped at the correct positions. An example of deconvolutions using the Chambolle-Pock algorithm is shown in figure 5.1 The example shown is a relatively ill-conditioned problem. We see that for low regularization there are still artifacts from the noise of the corrupted image, and for high regularization we get similar results as for the denoising examples shown in chapter 2. There are no noticable ringing effects, which we saw could be a problem in the end of chapter 3. The regularization parameter is on a different scale than for the TV denoising problem, with much smaller regularization parameters required for resulting regularization. We again see that choosing a suitable regularization parameter is crucial in order to obtain a good reconstruction.

5.2.4 Convergence criterion for numerical solvers

We will now discuss some aspects of implementing a Chambolle-Pock solver. We will only be using this algorithm for solving the TV problem. One of the main concerns of implementing a Chambolle-Pock solver is how to determine convergence. Effective determination of convergence deserves a study on its own, which is beyond the scope of this text. We will therefore attempt to quickly discuss some options and properties of the Chambolle-Pock algorithm.

We can determine convergence using the optimality conditions given by the Fenchel-

Rockafellar theorem. We have that the duality gap

$$\delta(u,p) = f(u) + g(Du) + g^*(p) + f^*(-D^*p) \tag{5.24}$$

$$= \frac{1}{2}\|Au - v\|^2 + \lambda|Du| - \|v\|^2 + \frac{1}{2}\|v - A^{-*}D^*p\|^2 + i_{B_\lambda}(p), \tag{5.25}$$

which satisfies $\delta(u^*, p^*) = 0$ only if u^* solves the primal problem and p^* solves the dual problem. One of the main problems with using this optimality condition is that it is expensive to evaluate. The term $\|f - A^- * D^*p\|^2$ is especially costly and possibly numerically unstable depending on the forward operator A. After some experimentation, which is not included in this text for the sake of brevity, one finds that convergence is generally fast when the regularization term is small relative to the data discrepancy term, and slow when the regularization term is large relative to the data discrepancy term. If the regularization term is very large we are most likely applying too much regularization, which is not useful in this work.

The rate of convergence depends on the parameter σ and τ. Convergence towards the correct solution can only be ensured when

$$\sigma\tau \leq \frac{1}{\|D\|^2} \approx \frac{1}{8}. \tag{5.26}$$

To ensure this is satisfied we set $\sigma = \tau = 0.25$ for the remainder of this text.

Because we are mainly concerned with the primal variable u, a simpler convergence criterion is to terminate the iterations when

$$\|u_k - u_{k-1}\| \leq \texttt{tol} \tag{5.27}$$

for some tolerance `tol`. This is much more computationally efficient, but does not ensure the solution is correct. This criterion does not ensure convergence for the dual variable p. This is especially evident for high regularization, when the convergence is slow. This will cause some problems later, but in general we are not concerned with very regularized solutions. For this reason, we choose this convergence criterion over the duality gap converence criterion. In addition, we always terminate the iterations after 100 iterations.

Chapter 6

Parameter selection

So far we have discussed inverse problems, and the need for regularization, and we know how to solve quadratic variation and total variation problems. However, we do not know how to choose the regularization parameter. As we saw earlier, different choices for the regularization parameter can yield very different reconstructed images. We begin by rescaling the regularization problem

$$S(t)(v) = \arg\min_u tL(u, v) + (1 - t)R(u) \tag{6.1}$$

and looking for regularization parameters $0 \leq t \leq 1$. This is essentially the same problem as before, but with $\lambda = \frac{(1-t)}{t}$. We do this because it is easier to look for regularization parameters $t \in [0, 1]$ than $\lambda \in [0, +\infty]$.

A way of phrasing the parameter selection problem is to find

$$t_{\text{opt}} = \arg\min_t \frac{1}{2}\|u^\dagger - S(t)(v)\|^2 = \arg\min_t E(t), \tag{6.2}$$

where $S(t)(v)$ is the reconstructed image and $u^\dagger$ is the original non-corrupted image. Here we use a simple metric to measure distance between images. It is also possible to use other loss functions $E(t)$. The goal of parameter selection is often to achieve a good parameter selection "automatically", without the need for human intervention.

We will always rescale the data to have elements between 0 and 1, so we divide the image by the maximum value attainable, which is 255 for 8-bit images. This step is not necessary, but it is useful to have all data on the roughly the same scale.

A good parameter choice depends on the features of the original, non-corrupted image as well as the signal-to-noise ratio. For the regularization methods we are looking at, it is infeasible to choose the regularization parameter based on just information about the noise or the forward model. In order to obtain the best results we must choose the optimal regularization parameter individually for every observed data v.

We can not solve equation (7.1) directly, because it would require the knowledge of $u^\dagger$, in which case we have already solved the inverse problem perfectly.

Figure 6.1 shows the loss function $E(t)$ for the quadratic variation and total variation problem for a test denoising example. The test denoising example we will use is the Lenna image with added Gaussian noise as was shown in figure 3.9. The loss functions

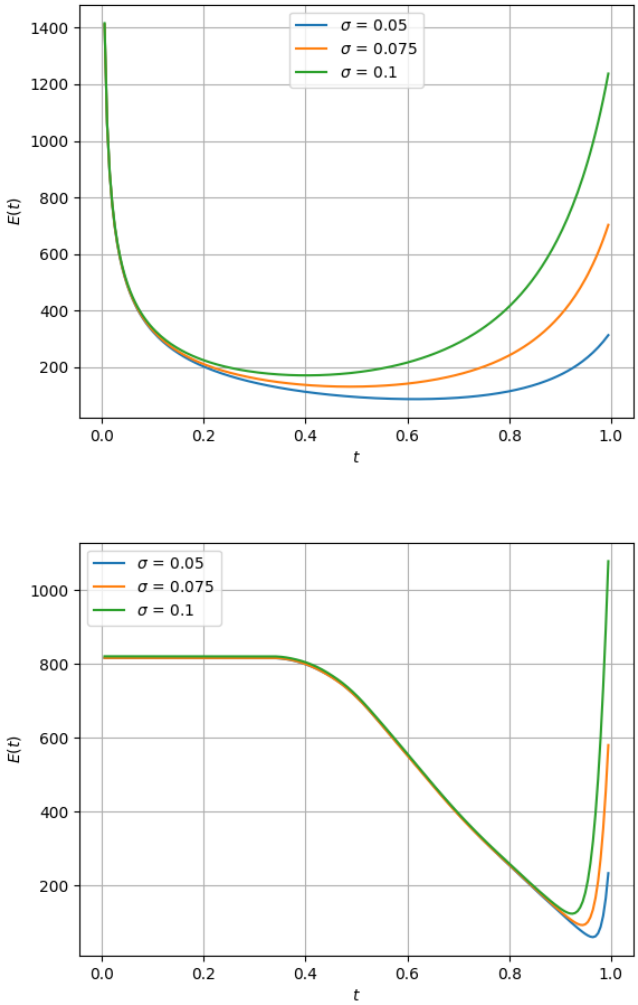

Figure 6.1: Parameter selection loss functions for the denoising problem on a 512×512 test image with Gaussian noise. The top image is created using QV denoising and the bottom one is created using TV denoising.

are continuous, with only one minimizer. It is important to note that the behaviour of $E(t)$ in practice depends on the numerical implementation of the solution operator $S(t)v$. The values of $E(t)$ for the TV problem for low t must be taken with a grain of salt, as the

numerical implementation struggles to converge here.

For the QV problem, the loss functions are convex and smooth, and does not have much curvature. For the TV problem, low t corresponds to the parameter values where TV yields a constant solution. The TV loss functions are unimodal, but not convex. The curvature around the optimal parameter for the TV problem is larger than for the QV problem. This means that a TV reconstruction is more sensitive to the parameter selection than the QV problem. We also see that the optimal TV reconstruction yields an image that is closer to the original image than the QV reconstruction.

6.1 Solving equations and optimization poblems of one variable

We will see that we will require methods for solving nonlinear equations

$$F(t) = 0, \tag{6.3}$$

and unconstrained optimization problems

$$t_{\text{opt}} = \arg\min_t G(t), \tag{6.4}$$

where $F : [0,1] \to \mathbb{R}$ and $G : [0,1] \to \mathbb{R}^+$. Therefore, we need to quickly discuss methods for solving such problems.

We will discuss two different types of methods, **bracketing** methods and **iterative** methods. We will need this for functions $F(t)$ and $G(t)$ that are computationally demanding to evaluate. In this case the work done by the solver is usually dwarfed by the work done by evaluating the function and its derivatives.

Bracketing methods work by finding smaller and smaller intervals where the solution lies. Iterative methods uses an initial guess to create a sequence of guesses that (hopefully) converge towards the desired solution.

We begin by looking at methods for solving equations. The simplest bracketing method for solving nonlinear equations is the **bisection method**. This method works if we can find two points t_l and t_r, $t_l < t_r$ so that $F(t_l)$ and $F(t_r)$ have opposite signs and the function $F(t)$ is continuous. The method works by bisecting this interval into two intervals, and the sign of the middle point $F((t_l + t_r)/2)$ will tell us if the solution lies in one interval or the other. We then either set This is useful if the function $F(t)$ is non-increasing or non-decreasing, or if we know that there exists a unique solution. In this case we can choose $t_l = 0$ and $t_r = 1$ and the method will surely converge to this unique solution, but it will in general converge to a solution.

One can easily show that for the bisection method, the k-th midpoint t_k satisfies

$$|t_k - t_{\text{sol}}| \leq \frac{|t_r - t_l|}{2^k}, \tag{6.5}$$

where t_{sol} satisfies $F(t_{\text{sol}}) = 0$. This is a convergence property that is very useful, as one can replace the left-hand side with the desired tolerance and solve the equation for n

to find how many iterations are required to achieve a desired tolerance. Additionally, the method only requires one function evaluation per iteration.

Iterative methods usually work using information about the derivative $F'(t)$ when it exists and is practically computable, and some approximation of the derivative otherwise. If $F(t)$ is differentiable, one of the most popular methods is Newton's method. In the cases we will look at, calculating $F'(t)$ is not always feasible. Because of this, we will instead use the **secant method**. A secant iteration is

$$t_{k+1} = t_k - F(t_k)\frac{t_k - t_{k-1}}{F(t_k)} - F(t_{k-1}). \tag{6.6}$$

The secant method requires two initial values t_0 and t_1, and requires only one function evaluation each iteration.

As a stopping criterion, we can stop the iterations either satisfy

$$|t_{k+1} - t_k| < \texttt{tol1}, \tag{6.7}$$

for some tolerance `tol1` or when

$$|F(t_{k+1})| < \texttt{tol2} \tag{6.8}$$

for some possibly different `tol2`.

The convergence of this method is slower than Newton's method. For similar conditions as the Newton method, one can show that it can converge with order equal to the golden ratio $\phi \approx 1.618$. While the convergence is slower, the fact that only one function evaluation is needed per iteration can make the secant method faster than Newton's method in practice.

Optimization of nonlinear functions can be challenging. The functions we will need to optimize are mostly continuous, and because we are optimizing over a compact subset we can be sure that a global minimizer exists [10]. Optimization problems are much easier if the function is unimodal, meaning the function is monotonically decreasing for $t \leq t_{\text{opt}}$ and monotonically increasing for $t_{\text{opt}} \leq t$ for some $t_{\text{opt}} \in [0, 1]$. t_{opt} is then the minimizer of the function. All convex functions are unimodal, but not all unimodal functions are convex.

A bracketing method for solving optimization problems is the **Golden section search** [27]. This method works for a unimodal function with the global minima inside a chosen interval $[t_l, t_r]$. The method works similarly to the bisection method where each iteration obtains a smaller interval wherein the solution lies. The Golden section search algorithm is stated in algorithm 1.

Similarily to the bisection method, one can show that the k-th iterate t_k satisfies

$$|t_k - t_{\text{sol}}| \leq \frac{|t_r - t_l|}{\phi^k}, \tag{6.9}$$

where $t_{\text{sol}} = \arg\min_t G(t)$ and $\phi = 1.618...$. While this convergence can be slow, it is predictable and easy to work with. It is possible to preselect the amount of iterations required to achieve a solution within some tolerance, which means the runtime of the algorithm is somewhat predictable.

Algorithm 1: Golden-section search algorithm.

Set $\texttt{tol} > 0$, $\texttt{maxiter} > 0$.
Set $\rho = \frac{3-\sqrt{5}}{2}$, $\phi = \frac{\sqrt{5}-1}{2}$
Set t_l and t_r, $\Delta t = t_r - t_l$
$\hat{t}_l = t_l + \rho\Delta t$, $\hat{t}_r = t_l + \phi\Delta t$
$g_l = G(\hat{t}_l)$, $g_r = G(\hat{t}_r)$
Set $i = 0$
while $i < \texttt{maxiter}$ **do**
  if $r_l < r_r$ **then**
    $t_r = \hat{t}_r$
    $\hat{t}_r = \hat{t}_l$
    $g_r = g_l$
    $\Delta t = \phi\Delta t$
    $\hat{t}_l = t_l + \rho\Delta t$
    $g_l = G(\hat{t}_l)$
  else
    $t_l = \hat{t}_l$
    $\hat{t}_l = \hat{t}_r$
    $g_l = g_r$
    $\Delta t = \phi\Delta t$
    $\hat{t}_l = t_l + \phi\Delta t$
    $r_r = G(\hat{t}_r)$
  end
  if $\Delta t < \texttt{tol}$ **then**
    return $(\hat{t}_r + \hat{t}_l)/2$
  end
end

A common iterative method for solving optimization problems are so-called **line search** methods [17]. Line search methods require a search direction, usually based on information about $G'(t)$ and $G''(t)$ and a step length criterion, whcih ensures that a step in the search direction is an improvement over the current step. Calculating a good step length can require many evaluations of $G(t)$ and possibly $G'(t)$, and can require problem-specific tuning in order to be effective. When $G'(t)$ and/or $G''(t)$ is not available they have to be approximated using for example finite differences. This means that each iteration requires at least two evaluations of $G(t)$, and potentially more to find a suitable step length.

6.2 Overview of existing parameter selection methods

We will now explain the methods for the general regularization problem

$$S(t)v = \arg\min_u tL(u, v) + (1 - t)R(u), \tag{6.10}$$

as well as how they specialize to the TV and QV problem. It should be noted that none of these methods are new, but our implementation of the methods are somewhat novel.

Existing methods for the parameter selection are mostly based on some well chosen heuristic criterion. Methods usually attempt to fulfill this criterion in two different ways: By solving the regularization problem and finding a parameter that satisfies the heuristic criterion at the same time or by solving the regularization problem for multiple regularization parameters and then choosing a solution that satisfies the criterion. We will focus on the latter approach.

Here we will assume a sequence of parameters $t_n = \frac{1}{1+t_0 q^n}$, $n = 0, ..., N$ for some $0 < q < 1$, $0 < t_0$ and integer N. The parameter selection problem can be solved by solving the inverse problem for each t_n and check if this reconstruction satisfies some criteria.

As we are presenting the methods we will illustrate them for denoising a 512×512 image with Gaussian noise and $\sigma = 0.1$.

The **Discrepancy principle (DP)** is based on formulating the regularization problem as

$$S(t)v = \arg\min_u R(u), \quad \text{so that } \|Au - v\|_2^2 \leq c, \tag{6.11}$$

for some well-chosen c [26]. In the case where the noise σ is known, it is possible to show that an upper bound for c is $\tau n^2 \sigma^2$, where τ is a tuning parameter which we will set to $\tau = 1$. The discrepancy principle can then be used to select a regularization parameter t_{opt} to satisfy

$$\|S(t)v - v\|_2 = \tau\sigma n. \tag{6.12}$$

We can then select the firs t_n that satisfies eq:DP within some tolarance.

We indeed see that if we insert $Au^\dagger$ for $S(t)v$ we have

$$\|Au^\dagger - v\|_2^2 = \sigma^2 \|w\|^2, \tag{6.13}$$

where the expected value of the right side is $\sigma^2 n^2$. Another way of thinking of it is if $\|AS(t)v - v\|_2^2 \geq \sigma^2 n^2$, we are most likely underfitting the solution, and if $\|AS(t)v - v\|_2 \leq \sigma^2 n^2$ we are most likely overfitting the solution. Figure 6.2 shows how $\|AS(t)v - v\|^2$ behaves as for an example problem. Something that is immediately obvious from the plots is that $\|AS(t)v - v\|_2^2$ is non-increasing and continuous in t. Conversely it is non-decreasing and continuous in λ. This was proven for the TV problem with injective A as seen in Lemma 2.3 in a work by Chambolle and Lions [6]. In fact, this can be proven for a general solution operator.

The **Quasi-Optimality criterion (QOC)** was first proposed by Tikhinov [23], where the parameter is selected to satisfy

$$\lambda_{\text{opt}} = \arg\min_\lambda \|\lambda \frac{\partial S(\lambda)v}{\partial \lambda}\|. \tag{6.14}$$

In most applications this is infeasible to calculate. In this case, we can choose a sequence $\lambda_i = \lambda_0 q^i$ for some initial guess λ_0 and $0 < q < 1$. In this case, we can approximate

$$\lambda_{\text{opt}} = \arg\min_\lambda \|\frac{\partial S(\lambda)}{\partial \lambda}\|_2 \approx \arg\min_{i \leq N_{max}} \|S(\lambda_i)v - S(\lambda_{i+1})v\|_2. \tag{6.15}$$

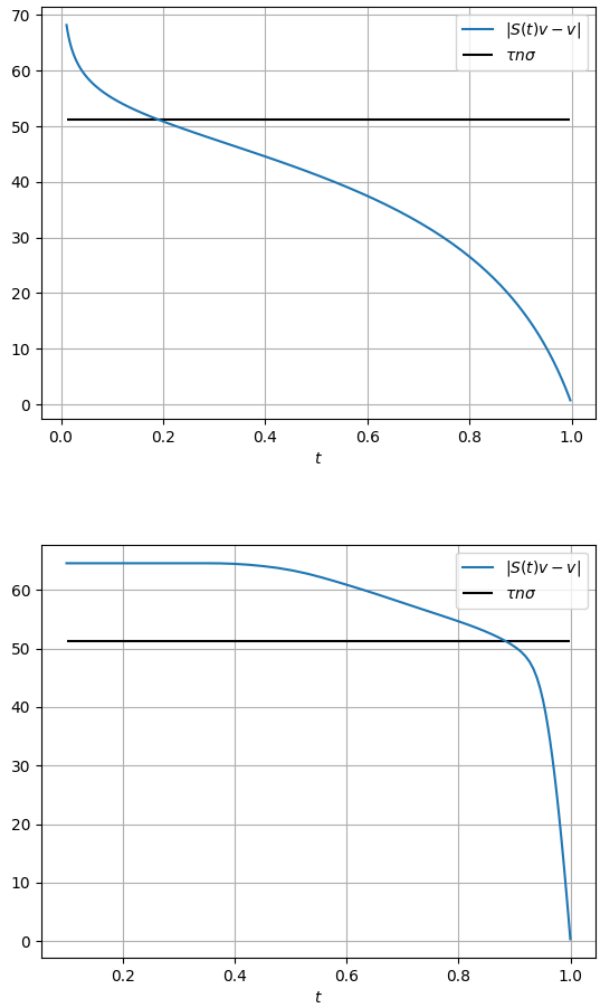

Figure 6.2: Illustration of Discrepancy principle for denoising. The top figure is created using QV regularization and the bottom figure is created using TV regularization respectively.

The rationale for this is that around the optimal parameter, small changes in the parameter usually does not change the solution of the problem drastically. One of the main advantages of this method is that it does not require the noise level σ. A study on applying the quasi optimality criterion to the TV problem in the infinite-dimensional case is done in a work by Kindermann et. al [13], where they suggest slight alterations to the quasi optimality criterion.

Figure 6.3 shows the functional in equation (6.15) for the example case and $\lambda_0 = 100$

and $q = 0.9$. From the figure some issues with the method are apparent. For both methods

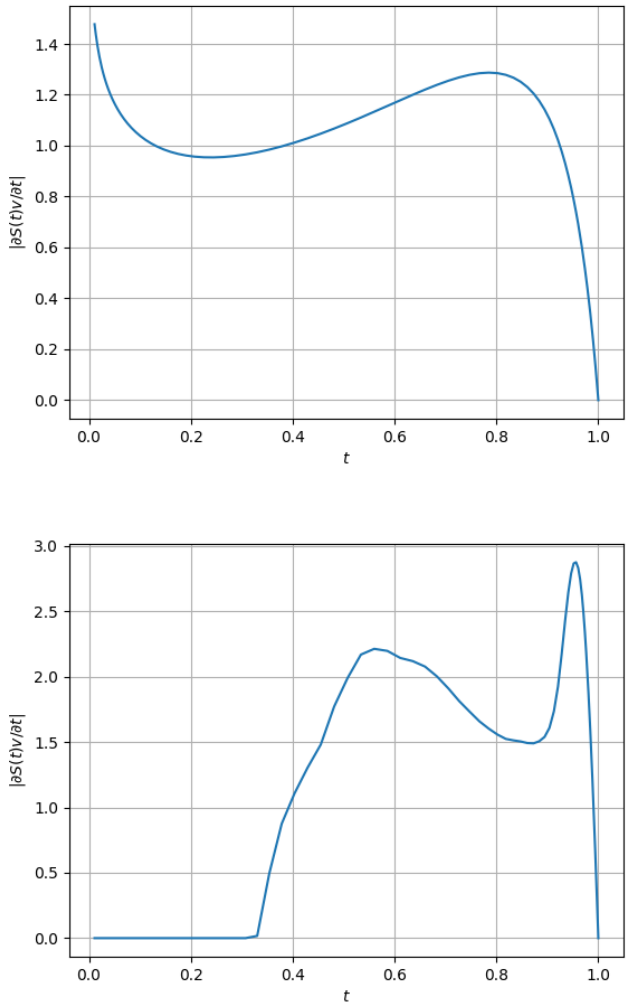

Figure 6.3: Quasi-optimality condition loss function. The top figure is created using QV denoising and the bottom figure is created using TV denoising.

a minimum is approached as t goes to 1, corresponding to no regularization. This is a problem for our interpretation of the method in finite dimensions. Furthermore, for the TV method, very large regularization also yields a minimum, as changing the regularization parameter does not change the solution operator here. For the TV problem, the parameter we are interested in is the local minimizer, not the minimizers close to $t = 0$ and $t = 1$. Similarily for the QV problem we are interested in the local minimizer, not the minimizer

near $t = 1$.

This means that choosing the initial guess, number of iterations and the parameter q is crucial for selecting a good parameter. Furthermore, it is hard to apply numerical methods to find the parameter as we did for the discrepancy principle. We have to evaluate the solution operator for a sequence of regularization parameter. For smoother functions like the QV problem, we can stop whenever the next iteration increases the QOC functional. In this case the method will stop either after one iteration, close to the desired solution, or at $t = 1$. For the TV problem, the QOC functional is so erratic that stopping early can yield bad results. In this case one needs to be extra careful with selecting parameters for the method. Compared to the other methods this is a major drawback which makes the method more computationally expensive and less robust.

The **L-curve method (LC)** uses the fact that a log-log plot of $R(S(t)v)$ against $L(S(t)v, v)$, usually has a horizontal part and a vertical part separated by a part of high curvature, resembling an L-shape [11][12]. A good parameter choice is then in the elbow of the curve. The reasoning for this is similar to the discrepancy principle. When the data discrepancy is large, we have an underfitted solution. After the elbow of the curve, the error reduces slowly, and we are overfitting. The L-curve method then chooses the parameter that maximizes the curvature of the L-curve.

If both $R(S(t)v)$ and $L(S(t)v, v)$ are twice differentiable with respect to t, it is possible to derive an analytic expression for the curvature of the L-curve, which then has to be optimized. This will in most cases be non-trivial, and possiblt unstable.

One of the proposed methods for finding the maximal curvature of the L-curve is [9]

$$t_{\text{opt}} = \arg\max_t L(S(t)v, v)R(S(t)v). \tag{6.16}$$

We can then choose the t_n that approximately solves (6.16).

Figures of the L-curves and the corresponding curvature for the example problem is shown in figures 6.4 and 6.5.

We see that the L-curves has a clear horizontal part, a part with high curvature and a vertical part. For TV regularization, there is also a point of high curvature corresponding to high regularization (low $R(S(t)v)$). This part is somewhat anomalous, and depends on how the numerical implementation handles high regularization.

As for curvature, we see that both QV regularization and TV regularization have unique global maximisers corresponding to the L-curve parameter choice. For TV regularization we see that there is a local minimizer between the L-curve parameter choice and large regularization. This is more or less profound depending on the numerical implementation of the regularization method. It is also known that the L-curve method can have this problem regardless of numerical implementation, and perhaps paradoxically, it only gets worse when there is less noise. This means that the L-curve method can potentially give an extremely regularized solution when almost no regularization is needed.

Generalized Cross-validation. Cross-validation is used to find the optimal parameter given a dataset of corrupted images and their true reconstruction. Generalized Cross-validation (GCV) is an extention of cross-validation that gives an estimate of the optimal parameter given a single corrupted image v. The estimate is the one that minimizes

$$GCV(t) = \arg\min_t \frac{\|AS(t)v - v\|_2^2}{|\text{trace}(I - AS(t))|^2}. \tag{6.17}$$

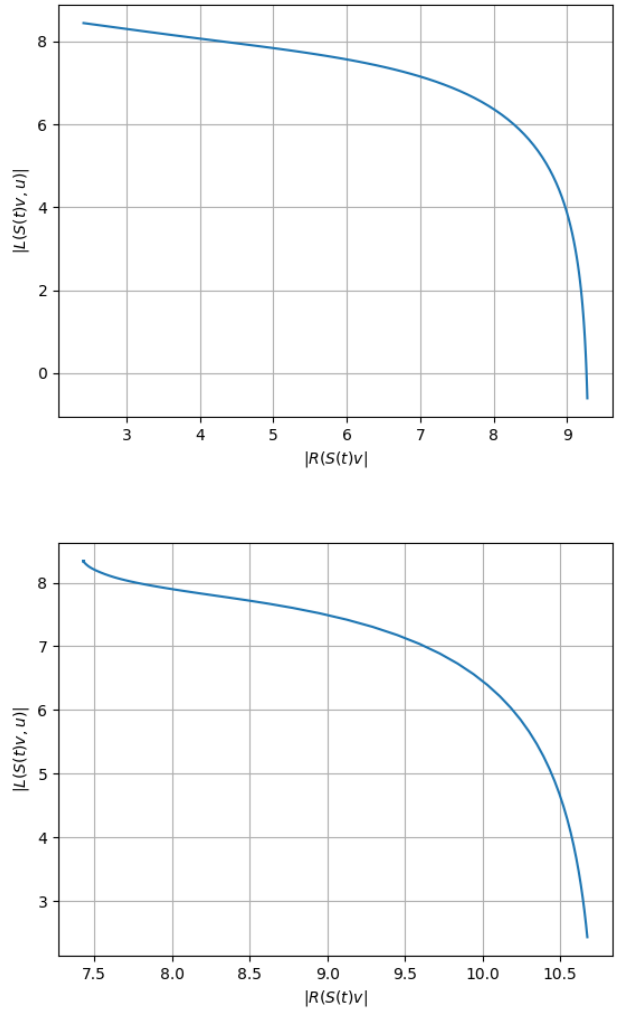

Figure 6.4: Log-log plot examplifying the L-curve for QV regularization and TV regularization respectively.

This trace in this formula can be difficult to compute, and impossible in most cases as the solution operator $S(t)$ is nonlinear. This means we cannot apply this method to TV regularization without doing some approximations or altering the method, but we can use it for QV regularization. A useful identity is

$$\text{trace} A = \sum_i \lambda_i, \tag{6.18}$$

where A is a linear operator and λ_i is the i-th eigenvalue. The eigenvalues of $AS(t)$ are

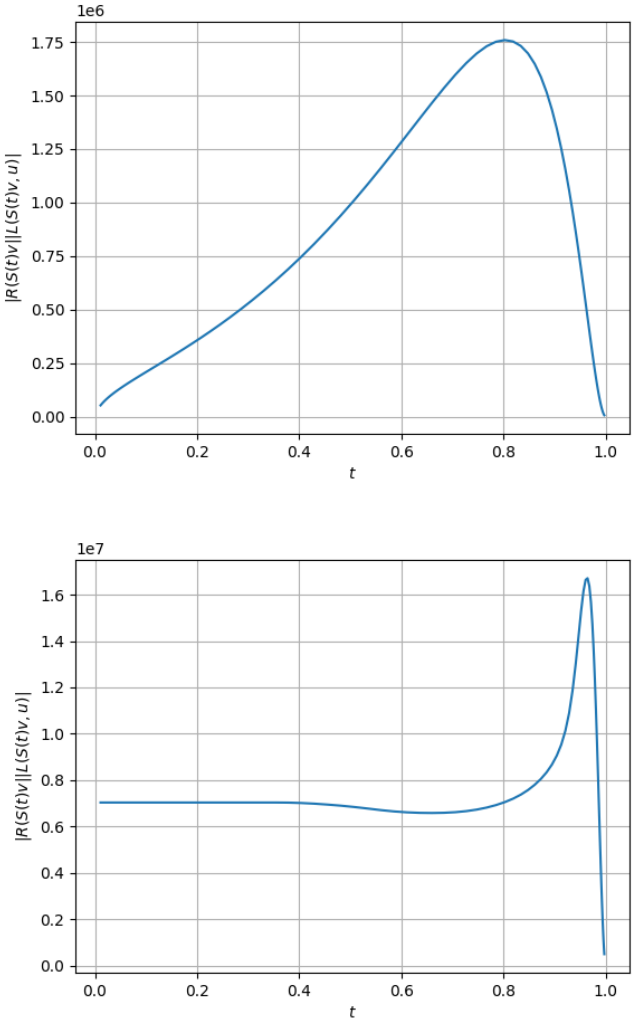

Figure 6.5: Curvature of the L-curve for QV regularization and TV regularization respectively.

not necessarily easy to calculate for general linear operators A and $S(t)$. In the case where they are both convolutions, we have that

$$\text{trace}(I - AS(t)v) = n^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} \text{DFT}(\kappa_A)\,\text{DFT}(\kappa_S), \tag{6.19}$$

where κ_A and κ_S are the convolution kernels of A and $S(t)$ respectively. For QV regular-

ization, we can find that

$$\mathrm{DFT}(\kappa_A)\,\mathrm{DFT}(\kappa_S) = \frac{|\,\mathrm{DFT}(\kappa)|^2}{|\,\mathrm{DFT}(\kappa)|^2 - \mathrm{DFT}(\kappa_L)}, \tag{6.20}$$

where κ is the convolution kernel corresponding to the linear operator A and κ_L is the discrete laplacian operator

$$\kappa_L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{6.21}$$

Figure 6.6 shows $GCV(t)$ for the test problem. We see that the function is similar to the

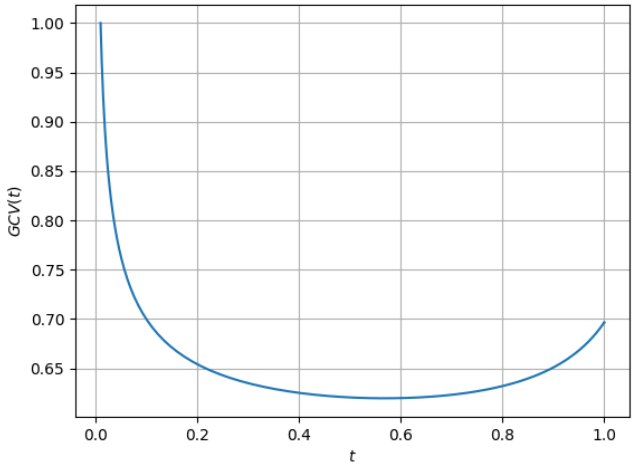

Figure 6.6: $GCV(t)$ using the example problem created using QV denoising.

parameter loss function $E(t)$, and is a function suitable for optimization. This is generally the case for different noise levels σ. A problem with the GCV function is that for t close to 0, corresponding to little regularization. In this case, evaluating $GCV(t)$ is numerically unstable.

6.3 Numerical solvers applied to existing methods

The recasting of regularization parameter from $\lambda \in \mathbb{R}$ to $t \in [0, 1]$ is useful for quickly finding a parameter satisfying a heuristic criterion. Instead of preselecting a sequence of parameters t_n we can more cleverly select a sequence of parameters through numerical methods. Remember, the main goal is to have a parameter selection that is as automatic as possible, without the need for human decision making.

For the discrepancy principle, it is clear that regularization problems have properties that make them suitable for root-finding algorithms. We can apply the bisection method

and similar method, and we can also apply the secant method and in rare cases, newton's method. For TV regularization, the solution operator is not differentiable, and newton's method is infeasible. For QV regularization, we can calculate the derivative, and therefor use Newton's method, but this will only be useful if precision of the heuristic method is of concern. Heuristic methods are often biased and innacurate, so finding a parameter that satisfies the heuristic criterion within some relatively low tolerance is often sufficient. The reduced convergence rate of the secant method is not of much concern as it is a much more practical method in this case.

For the L-curve, QOC and GCV methods, an optimization problem must be solved. As discussed, these three methods have their own quirks which can make the resulting optimization problem difficult. We will quickly discuss how to implement numerical solvers for these methods.

For the QOC method, not much can be done to improve the implementation. The clue is to have a good initial guess, and stop the iterations if no new minimizers have been found in a number of steps.

The L-curve is a bit more interesting. The objective is to find the maximal curvature, which was shown in figure 6.5. We can assume that $R(S(t=0)v)L(S(t=0)v,u) = R(S(t=1)v)L(S(t=1)v,u) = 0$, at least analytically. Something we can use here that we will use later is that the maximiser is "close" to $t = 1$. If an iterative method with or without a line search starts near $t = 1.0$ we should end up close to the desired solution. For bracketing methods like the golden-section search, we have similar problems as for QOC in that we need a good inital interval $[t_l, t_r]$ to search for a solution. We can safely choose $t_r = 1.0$ and choose t_l "close" to the maximal curvature.

For GCV, the main problem is the numerical instability around $t = 1$. This is again problematic in settings where very little regularization is required, which is also problematic for the L-curve method. For bracketing methods, we can avoid this problem by choosing the initial iteration (or right boundary of initial interval for bracketing methods) $t_0 = 1 - \varepsilon$ for some $\varepsilon > 0$. If the next iteration is somewhere to the right of this point, we stop the iteration.

For the L-curve and GCV method, we found Golden-section search to be more reliable for different problems than line-search methods, so we will apply Golden-section search to the L-curve and GCV methods. However, we predict that given more work, suitable line-search methods can be implemented that yield more computationally efficient results.

6.3.1 Testing the existing methods

We will now test the different parameter selection method on a test denoising problem for both QV and TV denoising and different noise levels.

For all bracketing methods we choose the initial interval $(t_l, t_r) = (0.8, 1.0)$ for TV denoising and $(t_l, t_r) = (0.2, 1.0)$ for QV denoising and tolerance 10^{-4}. For the secant method we choose $t_0 = 1.0$ and $t_1 = 0.99$ and tolerance 10^{-4}. For QOC we choose $\lambda_0 = 0.25$ for TV denoising and $\lambda_0 = 50.0$ for QV denoising, $q = 0.9$, and stop the algorithm if no new minimiser has been found in 5 iterations. All methods are stopped if they do not find a solution in 100 iterations. The algorithms are implemented in Python using Numpy. For timing the methods, we take the average processor time over ten runs.

The algorithms are run on a laptop with an Intel Core i5-8250U CPU and 8 gigabytes of RAM.

The results of applying these methods to the test problem is shown in tables 6.1.

$\sigma = 0.05$				
Method	t	$\lvert t - t_{\text{opt}}\rvert$	$\text{PSNR}(u, u^\dagger)$	Time [s]
Noisy	–	–	26.04	–
Optimal	0.9646	0.0	33.35	–
DP Secant	0.9422	0.02238	32.30	1.69
DP Bisection	0.9421	0.02241	32.30	4.32
QOC	0.9460	**0.01866**	**32.52**	5.02
LC Golden-section	0.800	0.1643	27.11	9.34
$\sigma = 0.1$				
Method	t	$\lvert t - t_{\text{opt}}\rvert$	$\text{PSNR}(u, u^\dagger)$	Time [s]
Noisy	–	–	20.08	–
Optimal	0.9236	0.0	30.24	–
DP Secant	0.8764	0.04718	28.99	2.12
DP Bisection	0.8764	0.04718	28.99	4.45
QOC	0.8714	0.05223	28.84	3.63
LC Golden-section	0.9450	**0.02137**	**29.20**	7.22

Table 6.1: Parameter selection results for TV denoising.

$\sigma = 0.05$				
Method	t	$\lvert t - t_{\text{opt}}\rvert$	$\text{PSNR}(u, u^\dagger)$	Time [s]
Noisy	–	–	26.04	–
Optimal	0.6141	0.0	31.75	–
DP Secant	0.3773	**0.2367**	**30.40**	**0.25**
QOC	0.9985	0.3845	26.09	3.69
LC Golden-section	0.200	0.4136	28.12	5.55
GCV Golden-section	0.8439	0.2298	29.89	3.05
$\sigma = 0.1$				
Method	t	$\lvert t - t_{\text{opt}}\rvert$	$\text{PSNR}(u, u^\dagger)$	Time [s]
Noisy	–	–	20.08	–
Optimal	0.3998	0.0	28.81	–
DP Secant	0.1764	0.2234	27.34	**0.31**
QOC	0.2364	0.1635	28.05	2.86
LC Golden-section	0.5872	0.1873	27.91	3.87
GCV Golden-section	0.5684	**0.1685**	**28.08**	3.05

Table 6.2: Parameter selection results for QV denoising.

From the results we see that the secant method is more computationally efficient than the bisection method for finding the discrepancy principle parameter for TV denoising, and it is much more computationally efficient than the other methods. The time is somewhat

artificial, as it depends heavily on the tolerance of the numerical methods. The L-curve does not converge to a value inside the initial interval for low noise for neither QV nor TV denoising. The Discrepancy principle and quasi-optimality criterion yields similar results, expect for low-noise QV denoising where it does not find a minimizer. Although we cannot draw conclusions from these results alone, we see that the methods exhibit some bias. For example, the discrepancy principle tends to regularize too much, and the GCV tends to regularize too little. Although we have only shown results for a single image here, similar results are obtained using other 512×512 grayscale images.

From these results we also see that while the different parameter selection methods are able to yield relatively good parameters, there is still some room for improvement.

6.4 Different parameter selection loss functions

Until now we have only investigated the parameter selection problem as it was given in equation (7.1). It should be noted that this is equivalent to choosing the parameter that yields the best PSNR value. For images, and inverse problems in general, the "optimal" parameter depends on the desired outcome of the reconstruction. Describing the desired outcome can be difficult to describe mathematically. For example, when denoising natural images the optimal parameter should be chosen so that noise is removed without adding too much blur, especially near edges in the image. The PSNR value does not necessarily take this into consideration. An alternative measure for image similariy is the SSIM, which was discussed in chapter 2. The optimal parameter using the SSIM for TV regularization can be chosen as

$$t_{\mathrm{SSIM}} = \arg\max_t \left[\mathrm{SSIM}(S_{\mathrm{TV}}(t)v, u^\dagger)\right]. \tag{6.22}$$

Figure 6.7 shows the loss function in equation (6.22). From the figure it is clear that the SSIM loss function is suitable for optimization in this case similarly to the original loss function. We also see that for this problem, the loss function is similar to the original loss function shown in figure 6.1, and they have similar optimizers.

In this work, we will choose parameter based on PSNR. The point is that other loss functions, like SSIM, can be used to measure the success of a parameter choice. In the next chapter, we will discuss some new ways of doing parameter selection, where the loss function chosen will be of more importance.

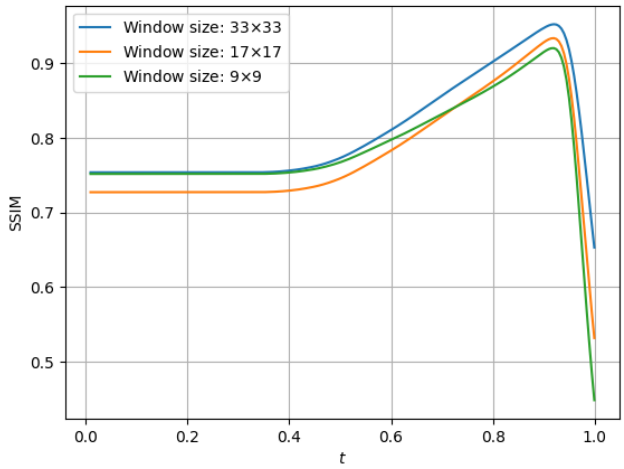

Figure 6.7: Loss function in equation (6.22) using different window sizes for the same noisy test image as the last section with $\sigma = 0.1$. For all window sizes, the optimal parameter is $t = 0.92$, which is roughly the same as the parameter which maximizes PSNR.

Chapter 7

A new parameter selection method

The method we will be investigating is inspired by work by Vito et al. [9]. In this paper, an unsupervised learning method for parameter selection of elastic net regularization was proposed and developed. The method is phrased in a statistical learning framework and uses results from high-dimensional probability theory.

Instead of solving the "true" parameter selection problem

$$t_{\text{opt}} = \arg\min_t \|S(t)v - u^\dagger\|^2 = \arg\min_t E(t), \tag{7.1}$$

we attempt to solve an approximate problem

$$\hat{t}_{\text{opt}} = \arg\min_t \|S(t)v - \hat{u}\|^2 = \arg\min_t \hat{E}(t) \tag{7.2}$$

where $\hat{u}$ is an approximate solution to $S(t_{\text{opt}})v$. To motivate why this can be a good idea, we consider our earlier example, and choose $\hat{u}$ as a Gaussian blurred image. From earlier, we know that a Gaussian filter can be used to remove noise. We choose the parameter for the Gaussian filter as the one that maximises PSNR. The resulting loss functions are shown in figure 7.1. We see that the minimizer of of the approximate loss function $\hat{E}(t)$ almost coincides with the minimizer of the true loss function $E(t)$. We can use this to solve the approximate problem instead of the true one, and use the parameter found here. We call this approach **Solution Regularization** for reasons that will become apparent. This of course requires access to an approximate solution. An approximate solution should be chosen from a different method, like another regularization method or a direct method that does not rely on parameter selection.

When the approximate solution $\hat{u}$ is given, we can find the parameter $\hat{t}_{\text{opt}}$ in essentially the same time as we can find the optimal parameter t_{opt} given the true data $u^\dagger$. In order for this to be effective, the alternate method should to have an easier and/or more computationally efficient implementation. The alternate method should also not yield On the other hand, one only needs the approximate loss function to have a minimizer that is close to the minimizer of the true loss function. This means that $\hat{u}$ does not in itself have to be a good reconstruction. It only has to be close to $S(t_{\text{opt}})v$ in the $\|.\|_2$-norm. We can interpret

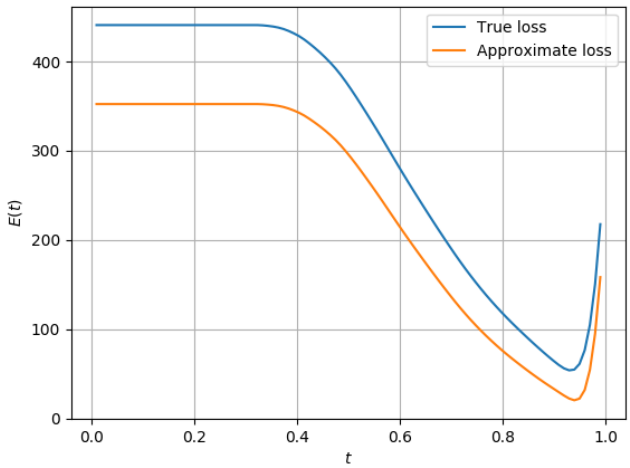

Figure 7.1: True and approximate loss functions from equations (7.1) and (7.2) respectively on a 512×512 noisy image with $\sigma = 0.1$. The approximate solution $\hat{u}$ is calculated using a Gaussian blur.

the method as finding a reconstruction that has the properties given by the solution operator $S(t_{\text{opt}})$ while being close to another data $\hat{u}$. We can thus interpret this as a kind of regularization on the reconstruction problem rewriting it to

$$u_{\text{opt}} = \min_{u=S(t)v} \|u - \hat{u}\|^2. \tag{7.3}$$

In other words, the method corresponds to minimizing a distance metric while regularizing based on some solution operator. Because of this reformulation, we will call the method **Solution Regularization** (SR). In general, this reformulation does not help us solve the problem, as the constraint is difficult to work with.

In the case where $\hat{u}$ is another regularization problem, we need to solve another parameter selection problem. As mentioned, this alternate parameter selection problem should be "easier", in the sense that it should be robust and computationally efficient.

Solving equations (7.1) and (7.2) was done using line search method in the work by Vito et al.[9], but we will instead use the Golden-section search, as we found it to give more reliable results. This means that in addition to finding an approximate solution $\hat{u}$, solving (7.2) is roughly as expensive as the L-curve method discussed in chapter 6.

In the referenced work, a specific choice for the approximate solution $\hat{u}$ was investigated, which we will discuss in an upcoming section. In this work, we will investigate more choices for $\hat{u}$, with the goal of applying it to TV regularization. We begin by investigating the properties of 7.2.

7.1 Parameter selection loss function

In order for the SR method to be effective, we need the empirical loss function

$$\hat{E}(\lambda) = \frac{1}{2}\|S(\lambda)v - \hat{u}\|_2^2 \tag{7.4}$$

to be a function suitable for optimization, perhaps for different choices of $\hat{u}$. The function should also be suitable for optimization for $\hat{u} = u^\dagger$, for the parameter selection problem to be well-posed. Through experimentation, one finds that for $\hat{u} = u^\dagger$, (7.4) has a unique global minimizer both for the QV and TV regularization in most realistic cases for images. Furthermore, for QV regularization the loss function seems to be convex. This is not the case for the approximate loss function for poorly chosen $\hat{u}$, which is something we will investigate.

In particular, we want to know if optimal points exist, if there exists local minima and if the function is convex, and how this depends on the solution operator $S(\lambda)v$ and approximate solution $\hat{u}$.

This analysis is easier for the solution operator $S(\lambda)$, not the rescaled one. We first need to be clear about what happens in $\lambda = 0$ and $\lambda = +\infty$, corresponding to no and infinite regularization respectively.

For both L_1 and L_2 regularization we have that

$$S(0)v = \arg\min u \frac{1}{2}\|Au - v\|_2^2, \tag{7.5}$$

which is just a least squares solution. In particular, if $v \in \mathrm{range}(A)$ this is the solution to $Au = v$. As for infinite regularization, we will assume that the solution operator is continuous in $\lambda = +\infty$, where

$$S(+\infty)v = \underset{u \in \ker(D)}{\arg\min} \frac{1}{2}\|Au - v\|_2^2. \tag{7.6}$$

For QV and TV regularization, an element in $\ker(D)$ is a constant vector. We generally want to avoid cases where there is either no regularization or infinite regularization. Both of these cases correspond to where the regularization function is not doing its job as intended.

If we assume the solution operator $S(\lambda)v$ is continuous in λ, which it is for both QV and TV regularization, minimizing the loss function (7.4) corresponds to minimization of a continuous function over a compact domain. In other words, a global minimizer exists.

If we assume $S(\lambda)v$ is differentiable, we have

$$E'(\lambda) = \langle S'(\lambda), S(\lambda) - \hat{u}\rangle, \tag{7.7}$$

and

$$E''(\lambda) = \langle S''(\lambda), S(\lambda) - \hat{u}\rangle + \|S'(\lambda)\|^2. \tag{7.8}$$

We can only safely assume that $S(\lambda)$ is differentiable for L_2 regularization. If we further assume that A and D have the same eigenspace, the solution operator satisfies

$$S'(\lambda)v = -(D^*D)(A^*A + \lambda D^*D)^{-2}A^*v, \tag{7.9}$$

and

$$S''(\lambda)v = 2(D^*D)^2(A^*A + \lambda D^*D)^{-3}A^*v, \tag{7.10}$$

which are both linear operators in v.

done

We can investigate the zeroes of $E'(\lambda)$ and the sign of $E''(\lambda)$ for the L_2 case. Initally, assume that $A^*A = D^*D = I$ (or just $A = D = I$), which yields

$$E'(\lambda) = -\langle (I+\lambda I)^{-2}A^*v, (I+\lambda I)^{-1}A^*v - \hat{u}\rangle = -\frac{1}{(1+\lambda)^3}\langle v, v\rangle + \frac{1}{(1+\lambda)^2}\langle v, A\hat{u}\rangle, \tag{7.11}$$

which is 0 only when

$$\langle v, A\hat{u}\rangle\lambda = \|v\|^2 - \langle v, A\hat{u}\rangle, \tag{7.12}$$

or for $\lambda \to +\infty$. The characteristics of this equation depends on the operator A. For images, we usually assume that v and $A\hat{u}$ have non-negative components, in which case the equation has a unique solution unless $A\hat{u} = 0$, which is why non-injective operators A can be problematic.

In this case, the minimizer of the loss function is

$$\lambda_{\text{opt}} = \frac{\|v\|^2}{\langle v, A\hat{u}\rangle} - 1, \tag{7.13}$$

when this is positive. If this is not positive, either $\lambda = 0$ or $\lambda = +\infty$ is the optimal solution. Here we see that the choice of $\hat{u}$ is crucial in order for the parameter selection problem to be easily optimizable. We also see that if the solution operator $S(\lambda)v$ is chosen poorly, the parameter selection problem can be hard to solve.

In the more general for operators A and D case we have

$$\begin{aligned} E'(\lambda) &= -\langle (D^*D)(A^*A + \lambda D^*D)^{-2}A^*v, (A^*A + \lambda D^*D)^{-1}A^*v - \hat{u}\rangle \\ E''(\lambda) &= \langle 2(D^*D)^2(A^*A + \lambda_{\text{opt}}D^*D)^{-3}A^*v, (A^*A + \lambda_{\text{opt}}D^*D)^{-1}A^*v - \hat{u}\rangle \\ &\quad + \langle (D^*D)(A^*A + \lambda D^*D)^{-2}A^*v, (D^*D)(A^*A + \lambda D^*D)^{-2}A^*v\rangle. \end{aligned}$$

Remembering that the function is continuous, if we can show that $E'(\lambda_{\text{opt}}) = 0$ and $E''(\lambda_{\text{opt}}) > 0$ we have shown that the only stationary point is λ_{opt}, and it is a global minimizer.

In particular, this is the case if A is a convolution operator and D^*D is the laplace operator, which is the case for QV regularization with certain boundary conditions. We can write

$$A^*A = Q^*\Sigma_A Q, \quad D^*D = Q^*\Sigma_D Q. \tag{7.14}$$

Here Q and Q^* are the forward and backward DFT matrices, and Σ_A and Σ_D are diagonal matrices with the eigenvalues of the operators. We know that the eigenvalues of A^*A and D^*D are the squared absolute value of their Fourier coefficients. Here we use Q so that it is uitary. We then have that

$$(A^*A + \lambda D^*D) = Q^*(\Sigma_A + \lambda\Sigma_D)Q, \tag{7.15}$$

is another convolution. Furthermore, $(\Sigma_A + \lambda\Sigma_D)$ is diagonal with only real, positive values. In this case $E'(\lambda)$ can be solved for many λ. In fact, it can be shown that $E'(\lambda)$ is a polynomial equation of order up to twice as large as the size of v and $\hat{u}$. We can then write

$$\begin{aligned} E'(\lambda) &= -\langle Q^*\Sigma_D(\Sigma_A + \lambda\Sigma_D)^{-2}QA^*v, Q^*(\Sigma_A + \lambda\Sigma_D)^{-1}QA^*v - \hat{u}\rangle \\ E''(\lambda) &= 2\langle Q^*\Sigma_D^2(\Sigma_A + \lambda\Sigma_D)^{-3}QA^*v, Q^*(\Sigma_A + \lambda\Sigma_D)^{-1}QA^*v - \hat{u}\rangle \\ &\quad + \langle Q^*\Sigma_D(\Sigma_A + \lambda\Sigma_D)^{-2}QA^*v, Q^*\Sigma_D(\Sigma_A + \lambda\Sigma_D)^{-2}QA^*v\rangle. \end{aligned}$$

Using the unitarity of Q, and the fact that diagonal matrices commute and are their own conjugate transpose when they contain only real values, we have that this is equal to

$$\begin{aligned} E'(\lambda) &= -\langle \Sigma_D(\Sigma_A + \lambda\Sigma_D)^{-3}QA^*v, QA^*v\rangle + \langle \Sigma_D(\Sigma_A + \lambda\Sigma_D)^{-2}QA^*v, Q\hat{u}\rangle \\ E''(\lambda) &= 3\langle \Sigma_D^2(\Sigma_A + \lambda\Sigma_D)^{-4}QA^*v, QA^*v\rangle - 2\langle \Sigma_D^2(\Sigma_A + \lambda\Sigma_D)^{-3}QA^*v, Q\hat{u}\rangle. \end{aligned}$$

Solving $E'(\lambda) = 0$ is difficult, and it is hard to see if this condition can be used to say anything about $E''(\lambda)$ in stationary points. For optimization purposes, we want $E''(\lambda) > 0$. While this is possible depending on v, $\hat{u}$ and A, the conditions they need to satisfy are not easily stated. We want to avoid technicalities, so we will instead try to intuitively argue when the loss function is suitable for optimization and when it is not.

Intuitively, if the parameter selection is well-posed for $\hat{u} = u^\dagger$, it should be well-posed for a $\hat{u}$ that is close to $u^\dagger$. However, it is clear that it is possible to choose $\hat{u}$ so that $E(\lambda)$ is non-convex and possibly not suitable for optimization. If $\hat{u}$ has large components compared to v, it is possible that $E''(\lambda)$ changes signs. We can choose: $\hat{u} = cS(\lambda')v$ for some λ' and large scalar c, and almost surely have $E''(\lambda) < 0$ for some λ.

For images we can assume that the approximate solution $\hat{u}$, the observed data v and the original data $u^\dagger$ have roughly the same norm. This means that the operator A should keep the pixel values at roughly the same sizes as before, which can always be satisfied by rescaling. Furthermore, we can assume that the solution operator $S(\lambda)$ does not drastically change the norm of the image. Remember, the reconstruction $S(\lambda)v$ should be close to the original data u. For images, the DFT components are usually large for low frequencies and small for high frequencies. The noisy image v will have larger high frequencies than the noiseless image u, and hopefully the approximate solution $\hat{u}$. This means that the first term of $E''(\lambda)$ should be larger than the second term.

If all of these assumptions hold, which they generally do for images, we can assume that the loss function is unimodal, and most likely convex. In fact, if we choose $\hat{u} = S(\lambda_{\text{opt}})v = Q^*(\Sigma_A + \lambda\Sigma_D)^{-1}A^*v$, we have that $E''(\lambda_{\text{opt}}) > 0$.

A case when things simplify is again when $A^*A = D^*D = I$, in which case we have $\Sigma_D = \Sigma_A = I$, which yields

$$E''(\lambda_{\text{opt}}) = \frac{3}{(1+\lambda_{\text{opt}})^4}\langle v, v\rangle - \frac{2}{(1+\lambda_{\text{opt}})^3}\langle v, \hat{u}\rangle, \tag{7.16}$$

If λ_{opt} is positive, we have

$$1 + \lambda_{\text{opt}} = \frac{\langle v, v\rangle}{\langle v, A\hat{u}\rangle} \tag{7.17}$$

by an earlier derivation, which means we have

$$E''(\lambda_{\text{opt}}) = \frac{\langle v, A\hat{u}\rangle^4}{\langle v, v\rangle^3} > 0, \tag{7.18}$$

and λ_{opt} is a unique minimzer.

For the L_1 case, things are much more difficult because the solution operator $S(\lambda)v$ cannot in general be written on a closed form. We believe unimodality of $E(\lambda)$ can be shown for TV regularization given some assumptions on v and $\hat{u}$, and possibly the linear operator A. This is something that should be investigated further.

Wit these results and our experimental findings, we have confidence in our propossed method.

7.2 Empirical estimator

In the referenced work [9] a specific choice for an approximate solution was investigated. We will explain some of the basics of this paper to motivate our work. The work uses results from high-dimensional probability theory to solve the paramater selection problem for finite-dimensional inverse problems. For a great resource on the topic of high-dimensional probability theory which is used in the referenced work see the comprehensive work by Vershynin [24].

The framework can be explained as follows: Assume that one has a dataset of corrupted data where the original data and the noise comes from independent sub-gaussian distributions

$$v^{(i)} = Au^{(i)} + \sigma w^{(i)}, \quad i = 1, ..., N. \tag{7.19}$$

All corrupted data comes from the same forward operator A and has the same noise level σ, but different noise realizations $w^{(i)}$. Let the non-centered covariance matrix of the original data u be Σ_{org}, which is a $n^2 \times n^2$ matrix. u and $w^{(i)}$ are here considered as random variables.

Assume that the covariance matrix satisfies $\text{rank}(\Sigma_{\text{org}}) = h$ for some $h << n^2$, and the forward operator A is injective on the range of Σ_{org}. We call the space spanned by this covariance matrix $\mathcal{V}$. This is a particular sparsity condition on the dataset, namely that the original data lies in a particular subspace $\mathcal{V}$. The first h eigenvectors then represent the information of the components of our dataset. This is similar to the ideas of principal component analysis.

The main idea is then to project the corrupted data onto this sparse subspace $\mathcal{V}$. This can be done with an orthogonal projection Π onto the first h eigenvectors of the covariance matrix. However, one rarely knows the covariance matrix of the original data. One can then use the empirical non-centered covariance matrix $\hat{\Sigma}$ which can be calculated from the dataset by

$$\hat{\Sigma} = \sum_i v^{(i)} v^{(i)T}. \tag{7.20}$$

The projection onto $\hat{\Sigma}$, $\hat{\Pi}$, can be used as an estimator for the projection Π. One can then show that $\hat{\Pi}$ is a good estimator of Π by leveraging high dimensional probability theory to

give probabilistic upper bounds on $\|\hat{\Pi} - \Pi\|$. A good approximate estimate should then be

$$A\hat{u} = \hat{\Pi}v. \tag{7.21}$$

We call this the **empirical estimator**. In the paper they instead define the estimator as $\hat{u} = A^{-1}\hat{\Pi}v$, and replace A^{-1} with the Moore-Penrose inverse if it does not exist. This method was shown to be a good parameter selection method for synthetic problems, including elastic net wavelet denoising. In cases where the forward operator A is orthogonal and $D = I$, the elastic net has a closed form solution. This can be used to show that the true loss function has has no local minima given some additional assumptions, and is suitable for optimization, similar to what we showed earlier for L_2 regularization. In the case of bounded Bernoulli noise, it is possible to find an analytic solution to the parameter selection problem, and an upper bound on $|t_{\text{opt}} - \hat{t}_{\text{opt}}|$. This gives some theoretical strength to the method.

There are some problems with this approximate solution, which we will discuss and try to alleviate. Firstly, we see that the calculating the empirical estimator in equation (7.21) requires solving a possibly ill-posed linear equation. Even if $\hat{\Pi}v$ is a good approximate estimate of Au, $\hat{u}$ can still be a bad estimate if A is ill-posed. As a way to alleviate this problem it is proposed in the paper that for non-injective A one uses the alternate loss functions

$$\hat{E}_P = \|PS(t)v - \hat{u}\|^2, \quad \hat{E}_M = \|AS(t)v - \hat{\Pi}v\|^2, \tag{7.22}$$

called the projected and modified loss function respectively. Here P is the orthogonal projection onto $\ker^{\perp}(A)$. These should stabilize the parameter selection somewhat. It was shown in the paper that while the empirical loss function in general gave poor results for non-injective A, the projected and modified loss functions gave better results.

If these still don't yield good results, one idea not discussed in the paper is to add regularization to either $\hat{E}_P$ or $\hat{E}_m$, but this could require another parameter selection.

The main problems with the empirical estimator for imaging are calculating the empirical estimator, and the assumptions made on the data. It is not clear what kinds of image datasets satisfies the sparsity conditions, or comes from some kind of statistical distribution in a meaningful way. For natural images, which we are working with, it seems highly unlikely that images follow some sparse distribution unless we are working with some very specific subset of images. Some potential cases are medical images or facial images with uniform backgrounds. However, in these cases the complexity is still high, meaning that the rank of the covariance matrix should still be high. The empirical covariance matrix $\hat{\Sigma}$ has maximum rank of the size of the dataset, given equation (7.20). To illustrate this, we need a suitable image dataset. For this cause, we will use the CIFAR-10 [15] dataset, which contains 60000 32×32 images divided into 10 different classes. Some example images from the "deer" class are shown in figure 7.2. The images are rescaled to have values between 0 and 1 and Gaussian noise is added, the result is shown in figure 7.3. For such

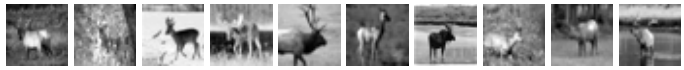

Figure 7.2: Example images from the "deer" class from the CIFAR-10 dataset.

Figure 7.3: Example images from the ”deer” class from the CIFAR-10 dataset with added Gaussian noise with $\sigma = 0.05$

small images, it i impossible to see the finer details of the image, but the images clearly contain deer-like animals with nature backgrounds. The question is, can we meaningfully extract the information of to dataset to find a sparse subspace? We select 512 such images to form a dataset. We calculate the ”true” (noiseless) and empirical (noisy) covariance matrices, and their eigenvalues. The result is shown in figure 7.4. From this figure we see that

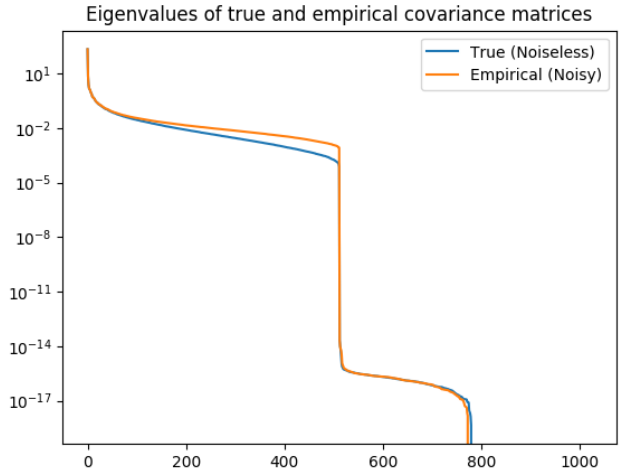

Figure 7.4: Eigenvalues of the true and empirical covariance matrices. Note the logarithmic scaling.

there is an initial fast decay in the size of the eigenvalues, followed by a slower decay until the 512-th eigenvalue where the rest of the eigenvalues are 0, ignoring round-off errors. We remind that the maximal rank of the covariance matrix is the size of the dataset, which in this case is 512. It is clear that the dataset does not satisfy the sparsity condition, as the eigenvalues until the 512-th eigenvalue are clearly non-zero, meaning we essentially have $h = 512$, which does not satisfy $h << n^2$. It is however clear that some eigenvectors carry more information than others, as one would also suspect. The empirical covariance matrix roughly has the same eigenvalues as the true covariance matrix adding σ. Figures 7.5 and 7.6 show the first ten eigenvectors (or ”eigen-deer”) of the true and empirical covariance matrices. It is clear that these few eigenvectors on their own do not carry much useful information about the dataset. We can now try to create empirical estimators for different values of h. We calculate the empirical estimators as the projection onto the first h eigenvectors of the ”true” covariance matrix Σ, calculated from the noiseless images, and the ”empirical” covariance matrix $\hat{\Sigma}$, calculated from the noisy images. The estimators are

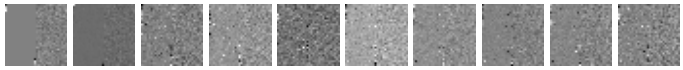

Figure 7.5: First 10 eigenvectors of the true covariance matrix.

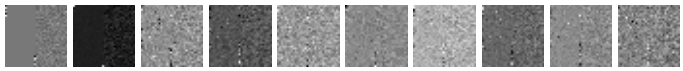

Figure 7.6: First 10 eigenvectors of the empirical covariance matrix.

calculated as

$$\hat{u} = P_h P_h^T v, \tag{7.23}$$

where P_h is a matrix containing the first h eigenvectors. We select the first image in figures 7.2 and 7.3 for an experiment. To evaluate how good these empirical estimators are we use the metric

$$\frac{\|\hat{u} - u^\dagger\|}{\|u^\dagger\|}, \tag{7.24}$$

where $\hat{u}$ is the empirical estimator and $u^\dagger$ is the original image. We try creating empirical estimators both for the noiseless and noisy images with both the true and empirical estimators for different choices of h, i.e how many eigenvectors we project the image onto. The results is shown in figure 7.7 Some example empirical estimators are shown in figure

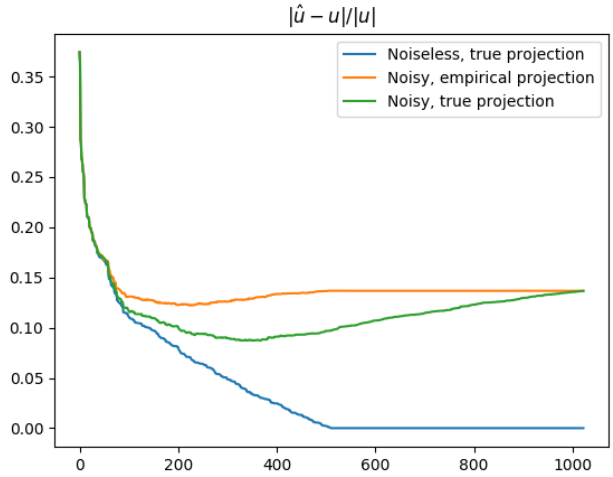

Figure 7.7: $\|\hat{u} - u\|/\|u\|$ for the different projections for noisy and noiseless images.

7.8, 7.9 and 7.10.

For the noiseless data, we see that we have a "perfect" empirical estimator for h larger than 512, as expected. For the noisy images, the empirical estimators deviate from the

Figure 7.8: Empirical estimators of an example noiseless image using true projection. From left to right, we have $h = 1, 16, 32, 128, 256, 512, 768, 1024$.

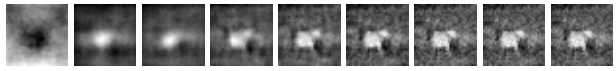

Figure 7.9: Empirical estimators for an example noisy image using empirical projection. From left to right, we have $h = 1, 16, 32, 128, 256, 512, 768, 1024$.

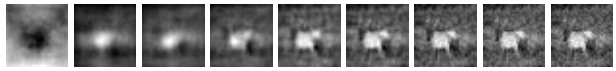

Figure 7.10: Empirical estimators for an example noisy image using true projection. From left to right, we have $h = 1, 16, 32, 128, 256, 512, 768, 1024$.

noiseless estimator, and actually become less close to the original image as h increases, before converging to a constant value which coincides with reconstructing the noisy image. The true projection yields better results than the empirical projection. It is worth noting that for the empirical projection, we just recreate the noisy image for $h > 512$. It seems like the "optimal" h, i.e the one that is closest to the original image, in the case of the empirical estimator here is $h = 221$. It is evident that for low h the empirical estimator is similar to a low-resolution or blurred reconstruction of the image, which we know can have noise-removing properties. As we add more and more eigenvectors to the projection we achieve finer details. Increasing the size of the dataset yields almost identical results. A larger dataset should yield a more accurate projection.

The main issue is how difficult it is to extract useful information from the dataset, and this is especially evident in images which have many non-linear features, in which case a linear projection is probably not optimal.

Calculating the empirical covariance matrix $\hat{\Sigma}$ and the empirical projection matrix $\hat{\Pi}$ is infeasible for larger images. For a dataset of $n \times n$ images the covariance matrix is a $n^2 \times n^2$ dense matrix. If we store this in 8-bit depth, for 512×512 images the covariance matrix requires ~ 69 gigabytes of memory. We then need to calculate the first h eigenvectors of this matrix, which is also tremendously expensive. The calculation required to do this method and the memory requirements make this infeasible for applications on bigger images.

In order to circumvent these issues we can downsample the dataset and do the projection on the downsampled images, which is less costly, and then upsample to achieve our empirical estimator. As discussed in earlier chapters, we will lose some information in the downsampling and upsampling process, especially if the downsampling is coarse.

From our earlier experiment it seems like natural images in general do not satisfy the sparsity conditions which the empirical estimators assume. It is however clear that projecting unto a lower dimensional subspace acquired from both the empirical (unsupervised) and true (supervised) covariance matrix can reduce the noise of the image. We have not

tested how suitable these empirical estimators are for parameter selection. Remember the end goal is to use the empirical estimator for parameter selection.

Then there is a final problem. In unsupervised cases, which is our aim, it is not unusual to have small datasets. In this case, even if the datasets satisfy some sparsity, we might not be able to create a good projection matrix because the rank of the covariance matrix is smaller than h. Because of the difficulty of extracting useful information from a dataset, we will instead be working on single images.

We will first consider a special case of the proposed method that is similar to working with single images. Consider a dataset containing the same original data $u^\dagger$, but with realizations of the noise. Thus every image satisfies $v(i) = Au^\dagger + \sigma w^{(i)}$. We can choose $h = 1$ for the empirical covariance matrix, as the sparse subspace is the space spanned by u. The covariance matrix then has one large eigenvalue and the rest of the eigenvalues are roughly equal to the noise level. This was investigated in the referenced work, and was shown to give good results. We will argue that in this case, the calculation of the empirical estimator simplifies, without the need for calculating the covariance and projection matrices.

We can first note that a good estimator for the denoised iamge is the mean of the observations

$$\bar{v} = \frac{1}{N}\sum_{i=1}^{N} u + \sigma w^{(i)} = u + \sigma\bar{w}. \tag{7.25}$$

Here $\bar{w}$ denotes the mean of the noise observations. It is well known that if $w^{(i)}$ are independent random variables with mean 0 and variance 1 we have

$$E(\bar{w}_i) = 0, \quad \mathrm{Var}(\sigma\bar{w}_i) = \frac{\sigma^2}{N}. \tag{7.26}$$

Thus, taking the mean gives us an approximate denoised image. With many noise models this can be used to measure the same signal many times and take the mean of the observations to reduce noise.

The covariance matrix can be written as

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(u+\sigma w^{(i)})(u+\sigma w^{(i)})^T = uu^T + \sigma(u\bar{w}^T + \bar{w}u^T) + \frac{\sigma^2}{N}\sum_{i=1}^{N} w^{(i)}w^{(i)T}. \tag{7.27}$$

If we assume that u and the noise vectors $w^{(i)}$ are linearly independent, the three first terms are rank 1 matrices and the last term is a rank N matrix. The covariance matrix has rank $N + 1$, meaning there are $N + 1$ non-zero eigenvalues. If we asssume for simplicity that $\|u\| = \|w^{(i)}\| = 1$ we see that the first term is larger than the other terms for small σ and large N. We see the eigenvector of uu^T is u with eigenvalue $\|u\|^2 = 1$. In fact, if we multiply the covariance matrix with the original data u, we have

$$\hat{\Sigma}u = u + \sigma(\bar{w} + \langle\bar{w}, u\rangle) + \frac{\sigma^2}{N}\sum_{i=1}^{N} w^{(i)}\langle w^{(i)}, u\rangle. \tag{7.28}$$

If we again interpret $w^{(i)}$ as random variables with mean 0 and variance 1 we have that the inner prodduct $\langle u, w^{(i)} \rangle$ satisfies

$$\mathrm{E}(\sigma \langle u, w^{(i)} \rangle) = \sigma \, \mathrm{E}(\sum_j u_j w_j^{(i)}) = 0, \quad \mathrm{Var}(\sigma \langle u, w^{(i)} \rangle) = \sigma^2 \sum_j u_j^2 = \sigma^2, \quad (7.29)$$

and is normal distributed.

It is worth noting that projecting a noisy image $v = u + \sigma w$ onto the true image u yields

$$\frac{\langle u, u + \sigma w \rangle}{\langle u, u \rangle} u = \frac{\|u\|^2 + \sigma \langle u, w \rangle}{\|u\|^2} u, \quad (7.30)$$

which is an unbiased estimator of u.

We can here apply some useful results from linear algebra: the sum of the eigenvalues of a matrix is equal to its trace, which is a linear operator, and the trace of an outer product between two vectors is the inner product between the vectors $\mathrm{trace}(uv^T) = u^T v$.

Thus we find that

$$\mathrm{trace}(\hat{\Sigma}) = \|u\|^2 + 2\sigma \langle u, \bar{w} \rangle + \frac{\sigma^2}{N} \sum_{i=1}^{N} \|w^{(i)}\|^2 = 1 + 2\sigma \langle u, \bar{w} \rangle + \sigma^2, \quad (7.31)$$

which is the sum of the eigenvalues. If we further assume that the vectors $u, w^{(1)}, ..., w^{(N)}$ are orthogonal, we see that the sum of the eigenvalues is $1 + \sigma^2$. Although orthogonality will not be the case in general, the inner products will be diminishingly small for images, so it is a useful simplification. be diminishingly small for images. In this case we also have that the mean of the observations $\bar{v} = u + \sigma \bar{w}$ is an eigenvector of the covariance matrix, which can be shown by inserting

$$\begin{aligned} \hat{\Sigma}(u + \sigma \hat{w}) &= u\|u\|^2 + \sigma u \langle u, \bar{w} \rangle \\ &\quad + \sigma(\sigma u \|\bar{w}\|^2 + u \langle u, \bar{w} \rangle + \sigma \bar{w} \|u\|^2 + \sigma \bar{w} \langle u, \bar{w} \rangle) \\ &\quad + \frac{\sigma^2}{N} \sum_{i=1}^{N} w_i \langle w_i, u \rangle + \sigma w_i \langle w_i, \bar{w} \rangle \\ &= (1 + \frac{\sigma^2}{N^2})(u + \sigma \bar{w}). \end{aligned}$$

The proportion of the eigenvalue $(1 + \frac{\sigma^2}{N^2})$ to the sum of all eigenvalues is

$$\frac{1 + \frac{\sigma^2}{N}}{1 + \sigma^2}. \quad (7.32)$$

This is the largest eigenvalue for large N and sufficiently small σ. Thus the mean is the largest eigenvector, which should be close to the original data $u^\dagger$. No covariance matrix or projection needed.

It could be possible to find the eigenvalues of $\hat{\Sigma}$ in the non-orthogonal case. The eigenvectors are in span$\{u, w^{(1)}, ..., w^{(N)}$, and it should not be impossible to do the same the

same analysis as above without the orthogonality assumption. However, even if orthogonality between the noise vectors and the data does not hold, the mean estimator $\bar{v}$ should still be a good estimator for u, especially when the noise level σ is not too high and N not too small.

We can apply this to larger 512×512 images. Calculating the covariance matrix and the largest eigenvector is infeasible in this case. Using the Lenna image with different noise levels $\sigma = 0.05, 0.075, 0.1$, the PSNR of the mean estimator for different N is shown in figure 7.11. We remind the reader that PSNR is on a logarithmic scale, and

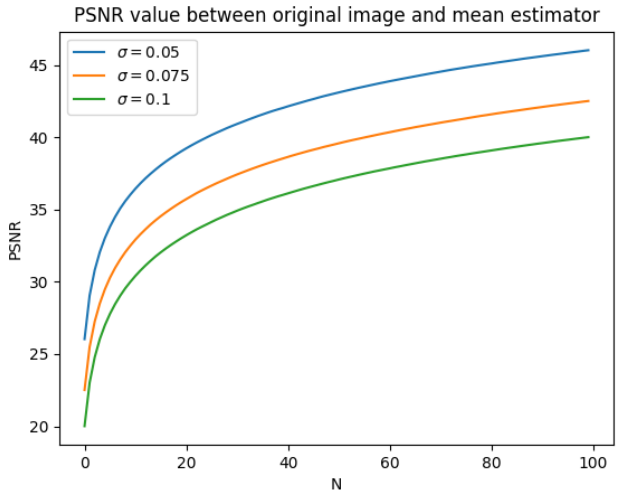

Figure 7.11: PSNR values of the mean estimator $\bar{v}$ for an increasing number of noisy images $v^{(i)} = u + \sigma w^{(i)}$ where $w^{(i)}$ is Gaussian distributed.

larger PSNR means the estimator is closer to the original image. It is clear that we can achieve arbitrary precision in our reconstruction as the number of data N increases. If we in addition attempt to project noisy data onto the original data we obtain PSNR values in the range $80 - 100$, which in this case is close to single floating point precision, and the projected image and the original image is completely indistinguishable by the human eye. This reinforces the fact that the inner product between an image u and a noise vector $w^{(i)}$ is diminishingly small.

The goal we set at the start was to use this for parameter selection by solving

$$t_{\text{opt}} = \arg\min_t \|S(t)v - \bar{v}\|^2. \tag{7.33}$$

It is worth noting here that $S(t)v$ is most likely a biased estimator of u. The goal the solution operator $S(t)v$ is then either to imbue the reconstruction with certain traits or to create a less noisy reconstruction of u. This method yields a good parameter for TV denoising for realistic values of N. We do not showcase the results here as we are mainly interested in single image denoising.

To summarize, we have investigated recent work on the topic of parameter selection [9]. Although the general method developed in this work does not apply well to natural images, it has potential in the case where one has a dataset with the same data, but different noise corruptions. In this case, we found that the method simplifies, and becomes computationally feasible. We will now develop these ideas further for single image reconstruction.

7.3 Single image reconstruction

We now want to apply what we have seen so far to single image reconstruction. In this case, we only have access to one image $v = u + \sigma w$. We will propose methods of resampling the image v to create a dataset $v^{(i)}$ based on the single observation v, similar to bootstrapping methods. We can then use this dataset to create an approximate estimator for the denoised image.

As discussed in chapter 4, image resampling can be done by interpolating the pixels of the image to create a continuous signal and sampling from this continuous signal. Two classes of image resampling are upsampling and downsampling, which creates a new image containing more or less pixels respectively. We saw that for simple methods, both these processes can be described by convolutions.

In our case, downsampling is the most useful, as we can downsample an image into N different images $v_{\text{ds}}^{(i)}$. We can then upsample these images to create a dataset of resampled images $v_{\text{us}}^{(i)}$. When we have achieved a dataset of resampled images, we can calculate the mean estimator

$$\bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_{\text{us}}^{(i)}, \tag{7.34}$$

and use this as an approximate denoised image. We will require that downsampled images are square images, which means that N can only be a square number. Applying the SR method yields

$$\hat{t}_{\text{opt}} = \arg\min_t \frac{1}{2}\|S_{\text{TV}}(t)v - \frac{1}{N}\sum_{i=1}^{N} S_{\text{us}}S_{\text{ds}}^{(i)}v\|^2. \tag{7.35}$$

Here $S_{\text{ds}}^{(i)}$ is a downsampling operator where the superscript denotes the grid used. The simplest downsampling algorithm to achieve this is nearest neighbour downsampling with $\alpha = 0.5$, which is what we will use. We can choose the grids so that

$$S_{\text{ds}}^{(1)}v = \begin{bmatrix} v_{1,1} & v_{1,3} & \cdots \\ v_{3,1} & v_{3,3} & \cdots \\ \vdots & \vdots & \end{bmatrix}, \quad S_{\text{ds}}^{(2)}v = \begin{bmatrix} v_{2,1} & v_{2,3} & \cdots \\ v_{4,1} & v_{4,3} & \cdots \\ \vdots & \vdots & \end{bmatrix}$$

$$S_{\text{ds}}^{(3)}v = \begin{bmatrix} v_{1,2} & v_{1,4} & \cdots \\ v_{3,2} & v_{3,4} & \cdots \\ \vdots & \vdots & \end{bmatrix}, \quad S_{\text{ds}}^{(4)}v = \begin{bmatrix} v_{2,2} & v_{2,4} & \cdots \\ v_{4,2} & v_{4,4} & \cdots \\ \vdots & \vdots & \end{bmatrix}$$

Figure 7.12: Image with Gaussian noise with $\sigma = 0.05$ downsampled into 4 different images using nearest-neighbour interpolation.

This can easily be extended to $N = 9, 16....$ Figure 7.12 shows a noisy image downsampled into four different images. We then upsample the images, take the mean and use this as an approximate solution in the solution regularization method. The result is shown in figure 7.13. We see that the average image removes some noise, and the remaining noise

Figure 7.13: To the left is the average image $\bar{v}$ of the upsampled images from figure 7.12. The upsampling algorithm used is linear interpolation. To the right is TV regularization using the parameter given in equation (7.35). The resulting parameter is $\hat{t} = 0.9746$, and the PSNR values of the two images are 31.97 and 32.70 respectively. Note that this is a better parameter than any of the other methods shown in table 6.1.

has some different characteristics from the original noisy image. The noise in the average

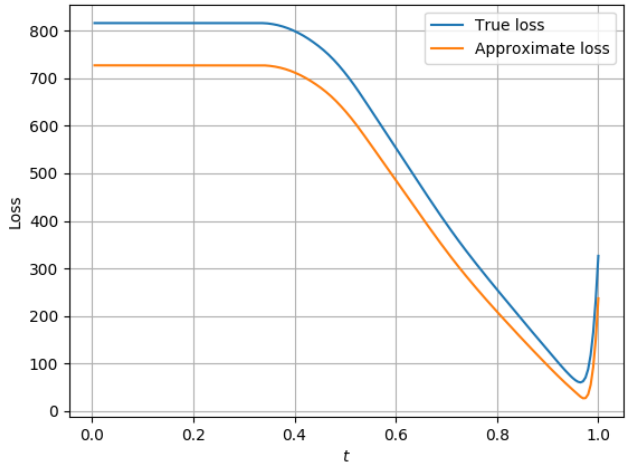

Figure 7.14: True and empirical loss functions from equations 7.1 and 7.35, where the approximate solution is the one given in figure 7.13.

image is more clumped together in local regions. The loss functions for the true and approximate parameter selection problems is shown in figure 7.14. We see that the two loss functions are very similar. The resulting TV regularization removes more noise, and has better PSNR value than the average image.

We can create a downsampled dataset of larger size by using coarser downsampling grids. However, these images quickly loose the details of the original image, especially if the original image has low resolution. Downsampling into more images will create a more blurred average image, but possibly remove more noise.

What we are doing when creating the average image is actually apply a linear operator

$$\bar{v} = \left(\frac{1}{N} \sum_{i=1}^{N} S_{\text{us}} S_{\text{ds}}^{(i)} \right) v, \tag{7.36}$$

which is a convolution. There are many different choices for N, S_{us} and $S_{\text{ds}}^{(i)}$, and the best choice depends on the noisy image v.

Experimentation shows that for 512×512 images, using neasest neighbour downsampling and bilinear or bicubic upsampling with $N = 4$, $N = 9$ or $N = 16$ always yields results comparable to the methods described in chapter 6. In particular, bicubic upsampling and $N = 9$ yields good results for noise values between $\sigma = 0.05$ and $\sigma = 0.1$. Some results are shown in table 7.1.

The parameters found using this method are better than the methods from chapter 6. Interestingly, the chosen parameter is too small for $\sigma = 0.05$, and too large for $\sigma = 0.075$ and onward, and almost perfect for $\sigma = 0.06$. This indicates that the optimal N and S_{us} differs depending on the noise.

σ	t_{opt}	$\hat{t}_{\text{opt}}$
0.05	0.9646	0.9597
0.06	0.9563	0.9552
0.075	0.9438	0.9479
0.09	0.9316	0.9403
0.1	0.9236	0.9350

Table 7.1: Results from parameter selection on a 512×512 noisy image with different noise levels σ using equation (7.35) with $N = 9$, nearest neighbour downsampling and bicubic upsampling.

The main problem with this approach is that there is that there is no automatic way of choosing N, S_{us} and $S_{\text{ds}}^{(i)}$, and the best choice depends on not only the noise level, but the resolution of the image and the contents of the image. In order to make this method robust one would either need to find N, S_{us} and $S_{\text{ds}}^{(i)}$ in a supervised setting, use some heuristic criterion on v or as we have done here: experiment until suitable settings are found. We would suspect that the settings we have chosen would generalize somewhat to other 512×512 images. This method will most likely fail if the image contains high frequencies, which would make it prone to aliasing in the downsampling and upsampling process.

The approximate solution $\hat{u}$ given by equation (7.36) is created using a particular convolution operator on the noisy image v. We showed as early as figure 7.1 that using a convolution with a Gaussian kernel could be used to achieve a good appoximate solution, but this requires that the Gaussian blur is created using a suitable parameter. There exists very limited literature on good parameter selection for denoising using Gaussian blur in the unsupervised case. However, setting this parameter to some default value, like $\sigma_{\text{blur}} = 1.0$, yields comperable results as those given in 7.1. Experimentation also shows that better results when σ_{blur} increases with the noise level σ. The point is that it is possible to do a good parameter selection when an approximate denoised solution is given by a linear operator, and the operator used can be the same, or tweaked slighlty, for different noise levels.

It is intriguing that we are able to select good parameters using downsampling and upsampling. This motivates us to investigate the properties of upsampling and downsampling further.

7.3.1 Upsampling and downsampling inverse problems

If we assume our inverse problem model,

$$v = Au + \sigma w \tag{7.37}$$

the noise w with noise level σ of our observed signal v is independent of the original signal u. In this subsection we interpret w as a random variable.

We have so far discussed downsampling which amounts to applying a linear operator S_{ds} to an image. Thus, when downsampling the observed signal v, we obtain

$$v_{\text{ds}} = S_{\text{ds}} v = S_{\text{ds}} A u + \sigma S_{\text{ds}} w. \tag{7.38}$$

In other words, the downsampled signal v_{ds} consists of the signal $S_{\text{ds}}Au$ and a noise vector $S_{\text{ds}}w$, which we will call w_{ds}, independent of $S_{\text{ds}}Au$. Thus we have

$$v_{\text{ds}} = S_{\text{ds}}Au + \sigma w_{\text{ds}}. \tag{7.39}$$

Say we want to solve this downsampled inverse problem instead. This is in fact a new inverse problem with a new forward operator $S_{\text{ds}}A$ and noise vector w_{ds}. Let us first investigate the noise vector $_{\text{ds}}$.

The expected value of the new noise vector is obviously 0, and variance of the new noise vector satisfies

$$(w_{\text{ds}}) = S_{\text{ds}}(w)S_{\text{ds}}^T. \tag{7.40}$$

This means that the downsampled noise vector does not necessarily have independent components. For Gaussian noise, w_{ds} is also Gaussian. The components of w_{ds} can have higher or lower variance (or noise level) depending on the interpolation method. We can write the new noise vector as

$$(w_{\text{ds}})_{i',j'} = \sum_i \sum_j \kappa(i' - \alpha i, j' - \alpha j) w_{i,j}, \tag{7.41}$$

where κ is the convolution kernel.

This also means that the different components of w_{ds} can have different noise levels. In particular, it is clear that

$$\text{Var}((w_{\text{ds}})_{i',j'}) = \text{Var}(w_{i,j}) \sum_i \sum_j \kappa(i' - \alpha i, j' - \alpha j)^2 = \sum_i \sum_j \kappa(i' - \alpha i, j' - \alpha j)^2. \tag{7.42}$$

This value depends on the grid chosen and the interpolation method. If the downsampled grid aligns with the original grid, the new noise vector will have independent componentss with the same noise level σ.

For interpolation kernels that attain negative values, like sinc and bicubic, the variance can actually increase, and noise is amplified. For bilinear interpolation the noise level will either be the same or lower.

We have thus found that downsampling can decrease the variance of the noise in the downsampled signal. In fact, if we know the downsampling algorithm, we know, with some work, the new variance of each pixel. With this knowledge, a possibility is to solve the regularization problem on the downsampled grid, and upsample the result and use this as an approximate solution. Furthermore, we can do this for N different downsampling grids, to essentially obtain N different approximate solutions, which we can then take the mean of. In the inverse problem model we described in chapter 2, we assumed the components of the noise vector was independent. For this reason, we will only use downsampling grids where the downsampled grid aligns with the original grid, meaning the components are still independent.

Upsampling has many of the same properties as downsampling, expect for upsampling there will always be new points which do not align with the original grid. The pixels that do not align with the original grid will not have noise independent of the other pixels, and can again have lower or higher variance based on the method. If we choose to denoise on the downsampled grid, it can be a wise decision to use an upsampling method that can

increase noise but better preserve the information in the image. After upsampling, we have N resampled images which we can take the mean of and use as $\hat{u}$. Even if we denoise on the downsampled grid, the upsampled images will not necessarily be good reconstructions on the original grid.

Note that the downsampled problem

$$v_{\text{ds}} = S_{\text{ds}}u + \sigma S_{\text{ds}}w, \tag{7.43}$$

is an inverse problem with forward operator S_{ds}. Instead of using one of the linear upsampling methods we have discussed we could use QV or TV regularization to calculate this upsampling.

Let us now discuss downsampling and upsampling for a convolution operator A. In this case, we can interpret the downsampled problem

$$v_{\text{ds}} = S_{\text{ds}}Au + \sigma S_{\text{ds}}w, \tag{7.44}$$

as a new inverse problem with forward operator $S_{\text{ds}}A$. We can calculate this operator by downsampling the kernel of the convolution A. Hopefully, $S_{\text{ds}}A$ is a better conditioned operator than A. If A is a low-pass filter, the downsampled operator $S_{\text{ds}}A$ will become more well-conditioned as it approaches the identity operator, which is much more well-conditioned.

Downsampling and upsampling can be beneficial because we can potentially solve a simpler parameter selection problem. Not only will the downsampled problem be less computationally intensive, but the properties of the donwsampled noise and forward operator can yield a better reconstruction.

As an example, we attempt to solve a denoising problem by downsampling, use one of the methods from chapter 6 to do parameter selection for TV denoising and upsample the result, then use this as an approximate solution. We downsample into N images, apply the method on one image and use the same parameter for the other $N - 1$ images. We upsample the images, take the mean, and use this as an approximate solution.

The approximate solution can be written as

$$\hat{u} = \frac{1}{N}\sum_{i=i}^{N} S_{\text{us}}S_{\text{TV}}(t_{\text{TV}})S_{\text{ds}}^{(i)}v. \tag{7.45}$$

Here the parameter t_{TV} has to be chosen using a parameter selection method. The L-curve method performed the best for $\sigma = 0.1$, so we apply this on the downsampled problem. Results are shown in figure 7.15. In this case, the results are good, surpassing the methods in chapter 6. Interestingly, the approximate solution from equation 7.45 is a good denoised image in its own right, and has essentially equivalent PSNR value as the resulting TV denoised solution. However, the two images have different denoising artifacts. The approximate solution still has some noise artifacts remaining, and the TV denoised solution has the artifacts one would predict from a TV denoised image.

This has some of the same problems as discussed in the previous section. We are left to choose N, S_{us} and $S_{\text{ds}}^{(i)}$, and we also require a parameter selection method on the downsampled images. This parameter selection is computationally cheaper than doing

Figure 7.15: The upper left image is a 512×512 image without noise. The upper right image is the same iamge with added Gaussian noise with $\sigma = 0.1$. The lower left is the approximate solution given by equation 7.45. N, S_{us} and $S_{\text{ds}}^{(i)}$ are chosen the same as in figure 7.12. The resulting image has PSNR value 30.19. The lower right image is the result from using the middle image as an approximate solution in the solution regularization method. The resulting parameter is $t = 0.9204$ and the resulting image has PSNR value 30.21.

parameter selection on the original noisy image. Particularly interesting here is how the approximate solution, which is much cheaper to produce than parameter selection on the original noisy image, outperforms the methods proposed in 6. As with the last section, this is something that should be investigated further.

7.3.2 QV regularization as approximate solution for TV regularization

TV regularization is better suited for image reconstrucion than QV regularization on most images of its edge-preserving properties. Parameter selection for the TV problem is thus the end goal. As we saw earlier, solutions to the QV problem are easier to calcuate, and the parameter selected is of less importance. While the TV solution operator is non-linear, the QV solution operator is linear, making analysis easier. Furthermore, the parameter selection problem for the QV problem is more stable, in the sense that a wider spread of parameters yield a decent reconstruction.

We can write equation (7.2) as

$$\hat{t}_{\text{opt}} = \arg\min_t \|S_{\text{TV}}(t)v - S_{\text{QV}}(t_{\text{QV}})v\|^2, \tag{7.46}$$

where t_{QV} is an approximate optimal parameter for the QV problem. Let us begin by choosing t_{QV} as the optimal parameter for QV regularization, i.e the one that minimizes PSNR. The resulting loss function is shown in figure 7.16. Again we see that the true loss

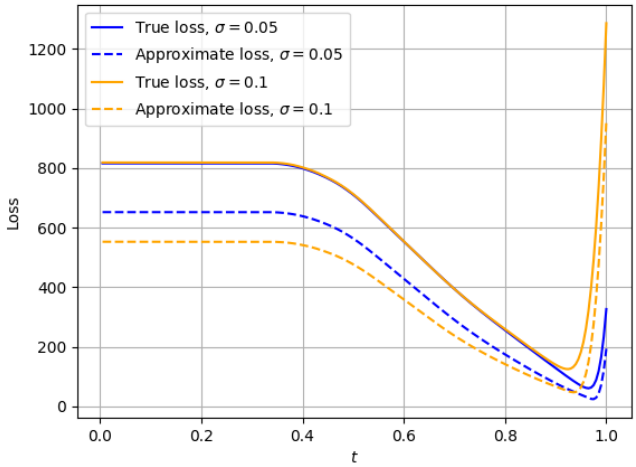

Figure 7.16: True and approximate loss functions from equations (7.1) and (7.46) for the denoising problem with different noise levels σ. The paramater t_{QV} is chosen to be the one that maximizes the PSNR value of $S_{\text{QV}}(t)v$.

function and approximate loss functions are very similar. However, this is not necessarily the case if the parameter t_{QV} is chosen poorly. From chapter 6 we saw that the GCV method gave the best results without requiring any knowledge about the noise level σ. We will therefore solve 7.46 with t_{QV} given by GCV. The results are shown in figure 7.17. In this case, the QV and TV denoising solutions are very similar, but the TV denoising solution is actually worse than the QV denoising solution in terms of PSNR. It should be noted that $S_{\text{TV}}(t)$ and $S_{\text{QV}}(t)$ yield similar reconstructions for t close to 1, corresponding

Figure 7.17: Top left is an 512×512 test image. Top right is a noisy image with Gaussian noise and $\sigma = 0.05$. Bottom left is a the result of applying QV denoising with parameter chosen by GCV. The resulting PSNR value is 29.89. Bottom right is the result of applying TV denoising with parameter chosen by equation 7.46 with approximate solution cohosen as the previous image. The resulting parameter is $t = 0.9878$ and PSNR value is 29.62.

with little regularization. In other words, if we want this approach to be successful, we need a better choice of t_{QV}, preferably one that regularizes too much instead of too little.

7.3.3 Parameter selection for deblurring

So far we have only tested parameter selection for denoising, and we will now apply the methods to deblurring.

The blur we will use is a 15×15 Gaussian kernel with $\sigma_b lur = 1.5$, and Gaussian noise with $\sigma = 0.02$. The condition number of the resulting blurring operator A is $\mathcal{K}(A) = 1.1 \times 10^9$. We will apply the methods to a 256×256 test image, because these methods are much more computationally intensive.

Firstly, the loss function is shown in figure We see that the loss function has a similar

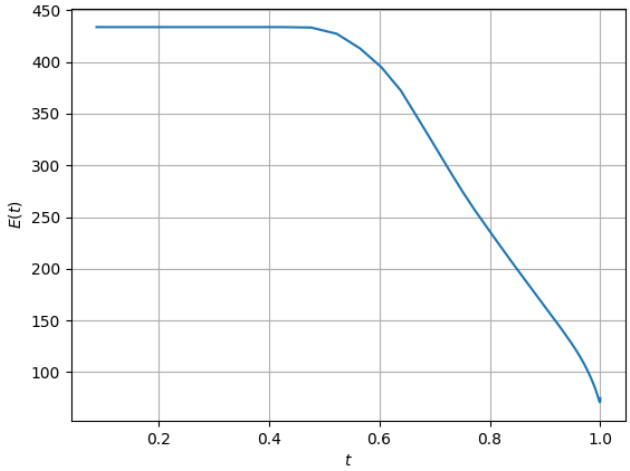

Figure 7.18: True loss function for the Total Variation deconvolution problem.

shape as before, but the minimizer is very close to $t = 1.0$, meaning that little regularization is required in order to achieve a good solution. Choosing a parameter that is too small can result in round-off errors, and possibly yield a bad solution. This can make parameter selection difficult, because we saw that some methods will yield parameters very close to $t = 1.0$ when they are essentially diverging. In fact, if we test the parameter selection methods for TV deblurring, this is exactly what happens for the Quasi Optimality Criterion and the L-curve method. The methods looks for parameters closer and closer to $t = 1.0$ until round-off errors happen, making them both very slow and unreliable. However, we have an additional method to work with for QV deblurring, which we will now test. Results for QV deblurring is shown in table 7.2. From the table we see that the L-

Method	t	PSNR
Noisy	–	25.60
Optimal	0.9718	27.70
DP	0.7817	26.38
QOC	0.9247	27.36
LC	1.0	6.30
GCV	0.9757	27.69

Table 7.2: Quadratic Variation deblurring results on test problem.

curve method has the same problems as for TV deblurring. It goes until a round-off error happens, and then erroneously chooses a poor parameter. The best method in this case is GCV, which yields near perfect results.

This motivates us to apply the idea from the last subsection, using a QV deblurred image as an approximate solution for the TV deblurring problem. The SR method yields

the parameter

$$\hat{t}_{\text{opt}} = \arg\min_t \|S_{\text{TV}}(t)v - S_{\text{QV}}(t_{\text{QV}})v\|^2, \tag{7.47}$$

where t_{QV} is selected using one of them methods from table 7.2. We compare these results to applying the discrepancy principle on the TV deblurring problem. The results are shown in table 7.3, as well as figures 7.19 and 7.20.

Method	t	PSNR
Noisy	–	25.60
Optimal	0.9979	28.21
DP	0.9804	27.00
SR QV QOC	0.9984	28.20
SR QV GCV	0.9998	27.76

Table 7.3: Total Variation deblurring results on test problem. Here SR QV QOC refers to solving equation (7.47) with t_{QV} chosen by the QOC and SR QV GCV refers to solving equation (7.47) with t_{QV} chosen by GCV.

From the results we see that solving (7.47) with the approximate solution given by QV deblurring yields better results than the discrepancy principle. In both cases, the TV reconstruction is a better reconstruction than the QV reconstructions used to choose the parameters. However, while the GCV method yields the best PSNR for QV reconstructions, the TV reconstruction using this t_{QV} in equation (7.47) barely improves the result, with the difference being indistinguishable by the human eye. These reconstructions clearly have noise artifacts in them. Using QOC for finding t_{QV} in equation (7.47) yields almost perfect results, with the TV reconstruction being slighlty sharper than the QV reconstruction. From this it is clear that choosing an approximate solution that maximizes PSNR does not give the TV reconstruction.

Although we have not included the timings in this experiment, finding a QV parameter is much more efficient than finding a TV parameter for the deblurring problem. When this parameter has been found, solving (7.47) is comparable in cost to the methods described in chapter 6.

We have also done some experiments with downsampling and upsampling deblurring problems, with promising results, but it will not be included in this text.

To summarize, we see that the SR method can give better results than the methods described in 6 when the approximate solution is well chosen. The main challenge of applying the method is then to choose this approximate solution robustly and efficiently. The results are promising for both denoising and deconvolution when the approximate solution is well-chosen.

7.4 Further work

In order to make the SR method more robust, it is natural to investigate the properties of approximate solutions and how they relate to the resulting parameter. It is worth investi-

Figure 7.19: Deconvolution results part 1. The upper left image is an 256×256 test image. The upper right image is a blurred and noisy image using Gaussian blur and Gaussian noise. The bottom left image is the optimal TV reconstruction. The bottom right image is the TV reconstruction using the discrepancy principle.

gating the parameter selection problem for a more general distance d,

$$t_{\text{opt}} = \arg\min_t d(S(t)v, u), \quad \hat{t} = \arg\min_t d(S(t)v, \hat{u}). \tag{7.48}$$

In this text we used the euclidean norm distance, but also mentioned the SSIM. It is possible that different distances can yield better reconstructions for different problems. It is then interesting to investigate how the true and approximate loss functions relate, and find how they differ depending on the approximate solution $\hat{u}$. The goal is to find theoretical bounds on $|t_{\text{opt}} - \hat{t}_{\text{opt}}|$. One should then look for good methods of producing $\hat{u}$ that minimizes $|t_{\text{opt}} - \hat{t}_{\text{opt}}|$. We saw that for denoising, choosing $\hat{u}$ using some linear denoiser gave decent results. Choosing $\hat{u}$ by QV denoising can potentially be a good option, but we would then need a good way of selecting the optimal parameter. Perhaps one can investigate the optimal operator for finding $\hat{u}$, by looking for the optimal solution operator

$$S_{\text{opt}} = \arg\min_S |\arg\min_t d(S_{\text{TV}}(t), u) - \arg\min_t d(S_{\text{TV}}(t), Sv)|. \tag{7.49}$$

Figure 7.20: Deconvolution results part 2. The upper left image is the QV reconstruction using the quasi optimality criterion. The upper right image is the TV reconstruction using the previous image as the approximate solution. The bottom left image is

Perhaps it is possible to choose $\hat{u}$ by some data-driven method. When a suitable distance d and approximate solution operator is found, solving the optimization problems 7.48 still has some room for improvement. In this text, we used the Golden-section search method, but we believe higher order methods like line search methods can be used. This can also be applied for the other methods discussed in chapter 6.

Lastly, while we have investigated this method for TV regularization for images, it should extend to other regularization methods in other settings. This is something which can be investigated further.

Chapter 8

Conclusion

The aim of this text was to investigate algorithms for image reconstruction modelled as an inverse problem. We have seen that regularization can effectively be used to solve inverse problems. In particular, total variation regularization can be effectively used to solve denoising and deconvolution problems for images. The accuracy of the reconstructions depends on the selection of a numerical parameter. Some existing methods were implemented using numerical methods. They gave decent results in some cases, but exhibited some room for improvement.

The main contribution of this work was the introduction of a new method for parameter selection, which we called Solution Regularization (SR). This method attempts to solve the parameter selection problem by using an approximate reconstruction, and find the parameter that yields the reconstruction that is closest to the approximate reconstruction. The main benefit of this method is that given an approximate reconstruction, the parameter can be found in approximately the same time as finding the optimal parameter when the original data is given. This requires solving an optimization problem, which we solved using the Golden-Section search method.

We investigated some choices for approximate reconstructions. We proposed finding an approximate reconstruction by applying other reconstruction methods, including other regularization methods requiring parameter selection. The idea is then to substitute a potentially difficult or computationally demanding parameter selection problem with a direct method or a simpler, less computationally demanding problem. This was shown to yield better results than existing method when the approximate reconstructions were well chosen. The main challenge with the SR method is choosing an approximate solution robustly and efficiently. The method is promising when this is the case, and good results were obtained for both denoising and deconvolution for images.

Further work will further assess how to find approximate solutions for total variation regularization and similar methods.

Bibliography

[1] One world imagine seminar, may 6 2020, 2020.

[2] P. Argoul. Overview of inverse problems. 2012.

[3] H. H. Bauschke, P. L. Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[4] A. K. Boyat and B. K. Joshi. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.

[5] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[6] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188, 1997.

[7] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

[8] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[9] E. de Vito, Z. Kereta, and V. Naumova. Unsupervised parameter selection for denoising with the elastic net, 2018.

[10] M. Grasmair. Minimisers of optimisation problems. 2017.

[11] P. C. Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM review*, 34(4):561–580, 1992.

[12] P. C. Hansen and D. P. O'Leary. The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM journal on scientific computing*, 14(6):1487–1503, 1993.

[13] S. Kindermann, L. D. Mutimbu, and E. Resmerita. A numerical study of heuristic parameter choice rules for total variation regularization. *Journal of Inverse and Ill-posed Problems*, 22(1):63 – 94, 2014.

[14] E. Kreyszig. Advanced engineering mathematics, 10th eddition, 2009.

[15] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[17] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[18] A. V. Oppenheim, J. R. Buck, and R. W. Schafer. *Discrete-time signal processing. Vol. 2*. Upper Saddle River, NJ: Prentice Hall, 2001.

[19] T. Sauer. Numerical analysis pearson addison wesley, 2006.

[20] C. Solomon and T. Breckon. *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[21] J. Stillwell. Logic and the philosophy of mathematics in the nineteenth century. In *Routledge History of Philosophy Volume VII*, pages 224–250. Routledge, 2013.

[22] E. Süli and D. F. Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[23] A. N. Tikhonov and V. B. Glasko. Use of the regularization method in non-linear problems. *USSR Computational Mathematics and Mathematical Physics*, 5(3):93–107, 1965.

[24] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[26] Y.-W. Wen and R. H. Chan. Parameter selection for total-variation-based image restoration using discrepancy principle. *IEEE Transactions on Image Processing*, 21(4):1770–1781, 2011.

[27] Wikipedia contributors. Golden-section search — Wikipedia, the free encyclopedia, 2020. [Online; accessed 2-July-2020].

[28] B. Wilhelm and J. B. Mark. Principles of digital image processing: core algorithms, 2009.

[29] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

Appendix A - Scaling of the Fourier transform

There exists several different conventions for the Fourier transform. Fourier series can be defined for periodic function u with period T

$$u(x) = \sum_{k=-\infty}^{\infty} C_k e^{\frac{i2\pi kx}{T}}. \tag{8.1}$$

In this thesis we chose to define the Fourier transform for frequencies ξ, but it can also be defined for angular frequencies $\omega = 2\pi\xi$ generalized to d dimensions as

$$\mathcal{F}(u(x)) = \int_{-\infty}^{\infty} u(x)e^{-i\omega x}dx, \quad \mathcal{F}^{-1}(U(\omega)) = \frac{1}{(2\pi)^d}\int_{-\infty}^{\infty} U(\omega)e^{i\omega x}d\omega. \tag{8.2}$$

With this formulation the Fourier transform is no longer unitary on L^2, i.e we no longer have$\mathcal{F}^* = \mathcal{F}^{-1}$. In order to preserve this property an alternate formulation exists:

$$\mathcal{F}(u(x)) = \frac{1}{(2\pi)^{d/2}}\int_{-\infty}^{\infty} u(x)e^{-i\omega x}dx, \quad \mathcal{F}^{-1}(U(\omega)) = \frac{1}{(2\pi)^{d/2}}\int_{-\infty}^{\infty} U(\omega)e^{i\omega x}d\omega, \tag{8.3}$$

which has the additional property of being more symmetric.

As for the DFT, the definition we used in this thesis is not unitary. The basis vectors of the transform are orthogonal, but not orthonormal. Orthonormality can be ensured by a simple scaling

$$\text{DFT}(u)_l = \frac{1}{\sqrt{2\pi}}\sum_{k=0}^{n-1} u_k e^{-i2\pi\frac{lk}{n}}, \quad \text{DFT}^{-1}(U)_k = \frac{1}{\sqrt{2\pi}}\sum_{l=0}^{n-1} U_l e^{i2\pi\frac{lk}{n}}. \tag{8.4}$$

Ludvigsen, Martin

Parameter selection for total variation regularization with applications in imaging

NTNU
Norwegian University of
Science and Technology