
Universität des Saarlandes, FR Informatik,
Cluster of Excellence,
Multimodal Computing and Interaction,
Probabilistic Machine Learning and Medical Image Processing

UNIVERSITÄT DES SAARLANDES

# Optimization Algorithms in the Reconstruction of MR Images: A Comparative Study

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science

von / by

Mateusz Malinowski

angefertigt unter der Leitung von / supervised by

Prof. Matthias Seeger
School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne, Switzerland

betreut von / advised by

Prof. Matthias Seeger

begutachtet von / reviewers

Prof. Matthias Seeger

Prof. Matthias Hein

Saarbrücken, June 9, 2011

**Non-plagiarism Statement** I hereby confirm that this thesis is my own work and that I have documented all sources used.

(Mateusz Malinowski)
Saarbrücken, June 9, 2011

**Declaration of Consent** I herewith agree that my thesis will be made available through the library of the Computer Science Department.

(Mateusz Malinowski)
Saarbrücken, June 9, 2011

# Acknowledgments

First and foremost, I would like to thank Prof. Matthias Seeger for giving me the opportunity to work on this exciting project. His advice and scientific experience helped me to give proper shape to this thesis. He also showed me how good research should work. Under his supervision I not only learned a lot but have also grown as a researcher.

I would like to thank all my lecturers in both Saarland University and University of Wrocław. Thanks to them every lecture that I took has influenced me in some specific way. Special thanks to Prof. Matthias Hein and Dr Simon Setzer for giving excellent lectures about convex optimization which helped me to understand some ideas from this broad field. Without their help it would have been much more difficult to finish this thesis.

Thanks also go to M2CI and IMPRS-CS for giving me the financial support and providing excellent environment to do research.

Many thanks Patrick Dempsey, the English teacher in Max-Planck-Institut für Informatik, who read this thesis and pointed language mistakes that I made.

I owe an enormous debt of gratitude to my friend Benjamin Roth who was always willing to help me and this thesis wasn't an exception. He did proofreading of most chapters of my thesis and gave many comments about language issues and the structure of the thesis. He also supported me many times during my stay in Germany.

My two flatmates Avishek Anand and Shanker Keshavdas helped me to smooth the language out and explained me the use of the articles 'the' and 'a'. Neither of these articles exist in my mother tongue and have always given me a lot of problems.

Many thanks also go to my flatmate Joanna Haręza who showed me that consecutive sentences must be semantically related to each other and shouldn't be treated as just a bag of facts. I hope that you will find time and energy to start your own scientific career.

Without my flatmates and friends from the university my life abroad wouldn't be so exciting. I would like to thank all of them for their time and help. Special thanks go to Gautham Adithya, Kamil Faber and Martin Suda for their great optimism and countless inspiring discussions.

I would like to thank my parents, brothers and grandparents for their never-ending support and understanding my decisions. Thanks to the sense of security that they have given to me I could have spent my time and energy on writing this thesis.

In my opinion presentation of the content and the content itself should go hand in hand. Therefore I would like to thank Dr Oliver Commowick for creating the latex template which I used in this thesis with minor modifications.

Last but not least, I want to thank my beloved girlfriend Joanna Pacia for her constant support and belief in me and my work. Without her help my stay abroad wouldn't be so easy. She also spent her precious time reading my thesis and pointing out mistakes that I made. Therefore I would like to dedicate my thesis to her.

# Abstract

Time that an imaging device needs to produce results is one of the most crucial factors in medical imaging. Shorter scanning duration causes fewer artifacts such as those created by the patient motion. In addition, it increases patient comfort and in the case of some imaging modalities also decreases exposure to radiation.

There are some possibilities, hardware-based or software-based, to improve the imaging speed. One way is to speed up the scanning process by acquiring fewer measurements. A recently developed mathematical framework called compressed sensing shows that it is possible to accurately recover undersampled images provided a suitable measurement matrix is used and the image itself has a sparse representation.

Nevertheless, not only measurements are important but also good reconstruction models are required. Such models are usually expressed as optimization problems.

In this thesis, we concentrated on the reconstruction of the undersampled Magnetic Resonance (MR) images. For this purpose a complex-valued reconstruction model was provided. Since the reconstruction should be as quick as possible, fast methods to find the solution for the reconstruction problem are required. To meet this objective, three popular algorithms *FISTA*, *Augmented Lagrangian* and *Non-linear Conjugate Gradient* were adopted to work with our model.

By changing the complex-valued reconstruction model slightly and dualizing the problem, we obtained an instance of the quadratically constrained quadratic program where both the objective function and the constraints are twice differentiable. Hence new model opened doors to two other methods, the first order method which resembles *FISTA* and is called in this thesis *Normed Constrained Quadratic FGP*, and the second order method called *Truncated Newton Primal Dual Interior Point*.

Next, in order to compare performance of the methods, we set up the experiments and evaluated all presented methods against the problem of reconstructing undersampled MR images. In the experiments we used a number of invocations of the Fourier transform to measure the performance of all algorithms.

As a result of the experiments we found that in the context of the original model the performance of *Augmented Lagrangian* is better than the other two methods. Performance of *Non-linear Conjugate Gradient* and *FISTA* are about the same. In the context of the extended model *Normed Constrained Quadratic FGP* beats the *Truncated Newton Primal Dual Interior Point* method.

# Contents

Chapter 1

# Introduction

## Contents



The structure of this brief introduction is as follows. We start from the general depiction of the chapter. Next, we briefly summarize each subsection. Finally, we introduce some notation exploited in the chapter.

Section 1.1 gives motivation which is behind the reconstruction of MR images. Moreover, a few concepts such as sparsity or compressed sensing are briefly shown. In Section 1.2, we cover basic definitions from convex optimization, and a few models that can be used in the reconstruction of MR images are briefly presented. One of these models is the primary model of this thesis. Section 1.3 gives references to previous work in the topic of sparse-reconstruction algorithms. Section 1.4 shows the contributions of this thesis. Finally, Section 1.5 lists all chapters in this thesis. It also gives a brief introduction to every chapter.

If $u = a + ib \in \mathbb{C}$ then $\Re u = a$ denotes the real part of $u$ and $\Im u = b$ denotes the imaginary part of $u$. Using this notation, we can also write $u = \Re u + i\Im u$.

We use notation $\boldsymbol{u}$ to denote a vector and it shouldn't be confused with notation $u$ used to denote a scalar. However, if $\boldsymbol{u} \in \mathbb{R}^n$ or $\boldsymbol{u} \in \mathbb{C}^n$ we use notation $u_j$ to denote the j-th entry of the vector. Notice that $u_j$ is a scalar in this case.

It is also assumed that indices start from zero. That is, if $\boldsymbol{u} \in \mathbb{R}^n$ then $u_0$ denotes the first element of the vector $\boldsymbol{u}$, $u_1$ denotes the second element of the vector $\boldsymbol{u}$ and so on. The last element of the vector $\boldsymbol{u}$ can be accessed by $u_{n-1}$.

## 1.1 Motivation

Imaging speed is an important factor in medical imaging. Usually, the longer the scanning duration, the more the image is prone to errors due to the patient motion. In addition, the patient comfort can be increased by speeding up the scanning process.

The imaging speed can be improved not only by constructing better hardware-based technologies but also by developing better reconstruction algorithms, for example we can reduce the scanning time by acquiring fewer measurements and then use them to reconstruct

the original image. This raises the problem of how the measurement and the reconstruction should be done in order to obtain the image which resembles the one obtained by taking all measurements.

Compressed sensing (CS) [Donoho 2006; Candés et al. 2006] is a mathematical framework which gives a partial solution to the posed problem. It shows that under certain conditions on the measurement matrix, the original vector that is assumed to be strongly sparse can be accurately recovered from a small number of measurements.

Vector is called strongly sparse if most of its coefficients vanish. However, the real data are often not strongly sparse. This leads to another way of expressing the notion of the vector sparsity, called weak sparsity. Vectors that are called weakly sparse or just sparse can be represented by a few significant non-zero coefficients. In this thesis images are represented by using the vector notation. Therefore the notion of the vector sparsity can readily be extended to the images.

A lot of natural and medical images are not sparse in the pixel representation. However, they can be sparse in other representations. Very often they exhibit the sparsity in a wavelet domain or under the finite difference operator, or in some other sparse-transform domain.

Transform sparsity is also widely used in image compression, for example JPEG and JPEG-2000 use discrete cosine transform (DCT) or a wavelet transform to map from the pixel representation to the sparse one. However, while compression algorithms exploit transform sparsity in the post-processing phase assuming that the acquisition was done before, the idea of compressed sensing is to use the property of being sparse already during the acquisition process.

The CS framework consists of three ingredients: a measurement matrix, a reconstruction model and the sparsity assumption. In the context of Magnetic Resonance Imaging (MRI) Lustig et al. [2007] proposed Variable density phase encoding and Seeger et al. [2009b] proposed Bayesian experimental design as the methods to construct the measurement matrix.

The reader can find more information about the sparsity and used models in the subsequent chapters.

## 1.2 Basic Definitions and Reconstruction Models

In this thesis two norms play the key roles. The first one is $l_1$-norm defined as follows

$$\forall_{\boldsymbol{u}\in\mathbb{C}^n} \; ||\boldsymbol{u}||_{l_1} := \sum_{j=0}^{n-1} |u_i| \tag{1.1}$$

The second norm is called $l_2$-norm and it is defined as follows

$$\forall_{\boldsymbol{u}\in\mathbb{C}^n} \; ||\boldsymbol{u}||_{l_2} := \sqrt{\sum_{j=0}^{n-1} |u_i|^2} \tag{1.2}$$

Frequently, however, working with the square of the $l_2$-norm is more useful

$$\forall_{\boldsymbol{u}\in\mathbb{C}^n} \; ||\boldsymbol{u}||_{l_2}^2 = \sum_{j=0}^{n-1} |u_i|^2 \tag{1.3}$$

Both $l_1$-norm and $l_2$-norm are special cases of the $l_p$-norm, where $p \in \{1, 2, \ldots\}$, defined as

$$\forall_{\boldsymbol{u}\in\mathbb{C}^n} \; ||\boldsymbol{u}||_p := \left( \sum_{j=0}^{n-1} |u_i|^p \right)^{\frac{1}{p}} \tag{1.4}$$

In this thesis we also use the TV-seminorm. Although different versions of a TV-seminorm can be proposed, we focus on the anisotropic version defined by

$$\forall_{\boldsymbol{u}\in\mathbb{C}^n} \; ||\boldsymbol{u}||_{TV} := ||\nabla_d \boldsymbol{u}||_{l_1} \tag{1.5}$$

where $\nabla_d$ is a finite difference operator[1].

In this thesis we deal with convex sets, convex functions and convex optimization problems. Therefore it is instrumental to mention their definitions. A set $\mathcal{D} \subseteq \mathbb{R}^n$ is called convex if and only if for every $\lambda \in [0, 1]$ and every $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{D}$ the following holds

$$\lambda\boldsymbol{u} + (1 - \lambda)\boldsymbol{v} \in \mathcal{D}$$

Consider a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ where $\overline{\mathbb{R}} := [-\infty, \infty]$ is the set of extended real numbers[2]. We also assume that the function is defined on every point in $\mathbb{R}^n$, however values $f(\boldsymbol{u}) = \infty$ or $f(\boldsymbol{u}) = -\infty$ are possible for some $\boldsymbol{u} \in \mathbb{R}^n$. We say that the function $f$ is convex if and only if for every $\lambda \in [0, 1]$ and every $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ the following holds

$$f(\lambda\boldsymbol{u} + (1 - \lambda)\boldsymbol{v}) \leq \lambda f(\boldsymbol{u}) + (1 - \lambda) f(\boldsymbol{v})$$

Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a convex function and let $\mathcal{F} \subseteq \mathbb{R}^n$ be a non-empty, convex and closed set[3]. By solving

$$\boldsymbol{u}^\star \in \arg\min_{\boldsymbol{u}\in\mathcal{F}} \; f(\boldsymbol{u}) \tag{1.6}$$

we mean finding $\boldsymbol{u}^\star \in \mathcal{F} \subseteq \mathbb{R}^n$ such that for every $\boldsymbol{u} \in \mathcal{F}$ the following (weak) inequality holds $f(\boldsymbol{u}^\star) \leq f(\boldsymbol{u})$. Then, we call $\boldsymbol{u}^\star$ a solution (or a minimizer) of problem (1.6), $f(\boldsymbol{u}^\star)$ a (local) minimum of $f$, $\mathcal{F}$ a feasible set and $f$ the objective function or the energy function. Moreover, the condition $\boldsymbol{u} \in \mathcal{F}$ defines constraints imposed on the vector $\boldsymbol{u}$. If $\mathcal{F} := \{\boldsymbol{u} \mid \forall_{j\in\mathcal{J}} \; h_j(\boldsymbol{u}) \leq 0\}$ where $\mathcal{J}$ is a set of indices (possible empty) and for each $j \in \mathcal{J}$ function $h_j(\cdot)$ is convex, then we say that problem (1.6) is a convex optimization problem. In this thesis, however, we often deal with an unconstrained optimization problem where $\mathcal{F} := \mathbb{R}^n$. In this case we sometimes write

$$\boldsymbol{u}^\star \in \arg\min_{\boldsymbol{u}} \; f(\boldsymbol{u}) \tag{1.7}$$

and assume that the dimensionality of the $\mathbb{R}^n$ space can be inferred from the context.

In the whole thesis, we assume that the objective function $f$ is a convex, continuous, proper[4] and coercive[5] function. These assumptions warrant that the minimum of $f$ exists

[1] Section 3.2 contains the definition of the finite difference operator used in this thesis.

[2] Arithmetic calculations involving both symbols $\infty$ and $-\infty$ the reader can find in Rockafellar [1997].

[3] Appendix A.2 in Boyd and Vandenberghe [2004] presents the concepts of open sets, closed sets and interior of a set.

[4] Function $f$ is proper iff. $\forall_{\boldsymbol{u}\in\mathbb{D}} \; f(\boldsymbol{u}) > -\infty \wedge \exists_{\boldsymbol{u}\in\mathbb{R}^n} \; f(\boldsymbol{u}) < \infty$.

[5] Function $f$ is coercive iff. $\lim_{||\boldsymbol{u}||_{l_2}\to\infty} \; f(\boldsymbol{u}) = \infty$.

and is unique, and that the set of minimizers is non-empty [Ekeland and Témam 1999]. More details about theory of convex optimization the reader can find in Rockafellar [1997], Ekeland and Témam [1999] and Boyd and Vandenberghe [2004].

There are a few models that can be considered in the reconstruction of MR images. One of the simplest models assumes that the signal is sparse in a wavelet-domain

$$\arg\min_{\boldsymbol{u}\in\mathbb{R}^n} \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \kappa||\boldsymbol{B}_a\boldsymbol{u}||_{l_1} \tag{1.8}$$

where $\boldsymbol{B}_a$ is an orthonormal wavelet operator, $\boldsymbol{X}$ is some measurement matrix and $\kappa$ is a model parameter (see Section 3.2 for more detailed definitions of $\boldsymbol{B}_a$ and $\boldsymbol{X}$).

We can also consider similar model with the TV-seminorm instead of the $l_1$-norm

$$\arg\min_{\boldsymbol{u}\in\mathbb{R}^n} \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \kappa||\boldsymbol{u}||_{TV} \tag{1.9}$$

where $\kappa$ is a model parameter.

A more sophisticated model includes both the $l_1$-norm wavelet regularization and the TV-seminorm

$$\arg\min_{\boldsymbol{u}\in\mathbb{R}^n} \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_{l_1} + \kappa_r||\boldsymbol{u}||_{TV} \tag{1.10}$$

where $\kappa_a$ and $\kappa_r$ are model parameters.

Although, in theory MR images are real-valued, in reality they may contain the imaginary part. This may happen due to the resonance frequency offsets, magnetic field inhomogeneities or eddy currents [Bernstein et al. 2004]. Thus it could be reasonable to switch from the real-valued models to the complex-valued ones and include a regularization term $\boldsymbol{B}_i$ that penalizes the imaginary part

$$\arg\min_{\boldsymbol{u}\in\mathbb{C}^n} \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_{l_1} + \kappa_r||\boldsymbol{u}||_{TV} + \kappa_i||\boldsymbol{B}_i\boldsymbol{u}||_{l_1} \tag{1.11}$$

where $\kappa_a$, $\kappa_r$ and $\kappa_i$ are model parameters. Moreover, the operator $\boldsymbol{B}_i$ should be defined in such a way that the following holds

$$||\boldsymbol{B}_i\boldsymbol{u}||_{l_1} = \sum_{j=0}^{n-1} |\Im u_i|$$

Although the last model (1.11) is defined over the complex-valued space, in this thesis we consider slightly modified version of this model defined over the real-valued space. Chapter 3 describes both the operators and the model used in this thesis in more details.

## 1.3 Related Work

Several first order methods have been developed to solve the reconstruction problem (1.8) and recently to deal with more complex models such as model (1.9) [Bioucas-Dias and Figueiredo 2007, 2008; Figueiredo et al. 2009; Alfonso et al. 2009; Beck and Teboulle 2009a,b; Yin et al. 2008; Goldstein and Osher 2009]. One of the difficulties that those methods face

is non-differentiability of the presented models. Another problem is non-invertibility of some operators such as $\boldsymbol{B}_i$. Some authors like Lustig et al. [2007] use a differentiable surrogate of the $l_1$-norm and employ standard methods such as the *Non-linear Conjugate Gradient* algorithm with backtracking line search to solve the reconstruction problem.

Although some earlier mentioned methods are quite general, it seems that direct approaches for solving problem (1.11) are needed in order to obtain the highest possible performance. In this thesis two such methods are presented. The first one is based on the method called *FISTA* [Beck and Teboulle 2009a,b] and the second one is based on *Augmented Lagrangian* [Nocedal and Wright 1999]. The latter was also used to derive *SALSA* [Figueiredo et al. 2009; Alfonso et al. 2009]. Also the equivalence was shown between the *Bregman* method and the *Augmented Lagrangian* method in Yin et al. [2008]. Moreover, at least one attempt has been made to exploit a second order method for solving problem (1.8) [Kim et al. 2007].

In this thesis we assume, for the sake of simplicity, that images are square [6].

## 1.4 Contributions

The main contributions of our work are:

- Three popular first order methods, *FISTA*, *Augmented Lagrangian* and *Non-linear Conjugate Gradient*, were adopted to work with model (1.11).
- The elastic extension of model (1.11) and its dual form was shown.
- Two methods, a first order method and a second order method, that can be used to solve the elastic extension of model (1.11) were presented. The first order method is called in this thesis *Normed Constrained Quadratic FGP*, and the second order method is the adaptation of *Interior Point* method to work with the elastic extension of model (1.11).
- The adopted *FISTA* and the adopted *Augmented Lagrangian* together with the *Non-linear Conjugate Gradient* method were compared on the basis of model (1.11) that arises during the reconstruction of MR images.
- *Normed Constrained Quadratic FGP* and the adopted *Interior Point* method were compared on the basis of the elastic extension of model (1.11).

## 1.5 Outline of the Thesis

Chapter 1 is an introductory chapter. It gives motivation to the study of the reconstruction of subsampled MR images. It also provides basic definitions of convex optimization theory, presents related work and shows main contributions of our work. Finally, it outlines the whole thesis. Chapter 2 focuses mainly on the representation of images used in the comparison.

[6] That is $n_x = n_y$ where $n_x$ and $n_y$ is, respectively, the number of columns and the number of rows of a given image.

It also concisely and intuitively explains the concept of sparsity and compressed sensing. In Chapter 3, we define linear operators that are used in this thesis and we give some insight into the $l_1$ reconstruction model used in the MRI setting. In Chapter 4, we define the system of linear equations called elastic linear system. A special case of this system occurs frequently in algorithms as a subproblem to solve. This special case can be solved efficiently by the method presented in that chapter. In Chapter 5, we present the concept of projection and its generalization called proximity operator. Both concepts play important roles in the reconstruction problem considered in this thesis. In Chapter 6, we present three first order methods used to solve the problem given in Chapter 3. That is, we briefly describe *Non-linear Conjugate Gradient* algorithm and we derive two other methods. One of them is based on *Augmented Lagrangian* approach and the second one is based on the method called *FISTA*. In Chapter 7, we introduce a slightly different model to the one introduced in Chapter 3. The new model was obtained by considering additional $l_2$-norm terms in the regularization part of the original model. Next, we dualize the problem which happened to be an instance of a quadratically constrained quadratic program. Finally, two algorithms are introduced that are suitable for solving this type of problems. The first algorithm is a first order method, whereas the second method is a second order method called *Interior Point*. In Chapter 8, we describe the experiments and show results of the comparison of the previously presented methods. In this chapter, we also discuss the results of the experiments. Finally, in Chapter 9, we briefly summarize the whole thesis and show some possible future directions.

Appendix A contains a list of notations used in this thesis. In Appendix B the reader can find a more precise explanation of some notational issues. In particular, it includes an explanation of the $\boldsymbol{u}^{\mathcal{M}}$ symbol that is used in some other chapters. Appendix C contains definitions that could be useful in understanding Chapter 6 and Chapter 7. However, readers familiar with theory of convex optimization may easily skip the material presented in Appendix C and regard it as a reference. Otherwise, a thorough reading of the definitions that this appendix contains is recommended. Appendix D presents supplementary proofs not included in the chapters. Appendix E defines the standard vectors. We use standard vectors in Section 6.1 and in Section 7.4 to derive the gradient or Hessian of an energy function. Appendix F briefly shows the differential calculus proposed by Minka [2001] and used in this thesis. The notion of derivatives is needed since some algorithms such as *Non-linear Conjugate Gradient* and *Interior Point* require the gradient or Hessian to be computed. Appendix G contains a few line-search methods that can be used in order to find a suitable step-size required by some algorithms. Appendix H shows additional figures obtained from the experiments.

Although it is recommended to read this thesis consequently starting with Chapter 1, there is the possibility to read the chapters in a different order. In order to avoid confusion the graph of the chapter dependency (see Figure 1.1) have been included.

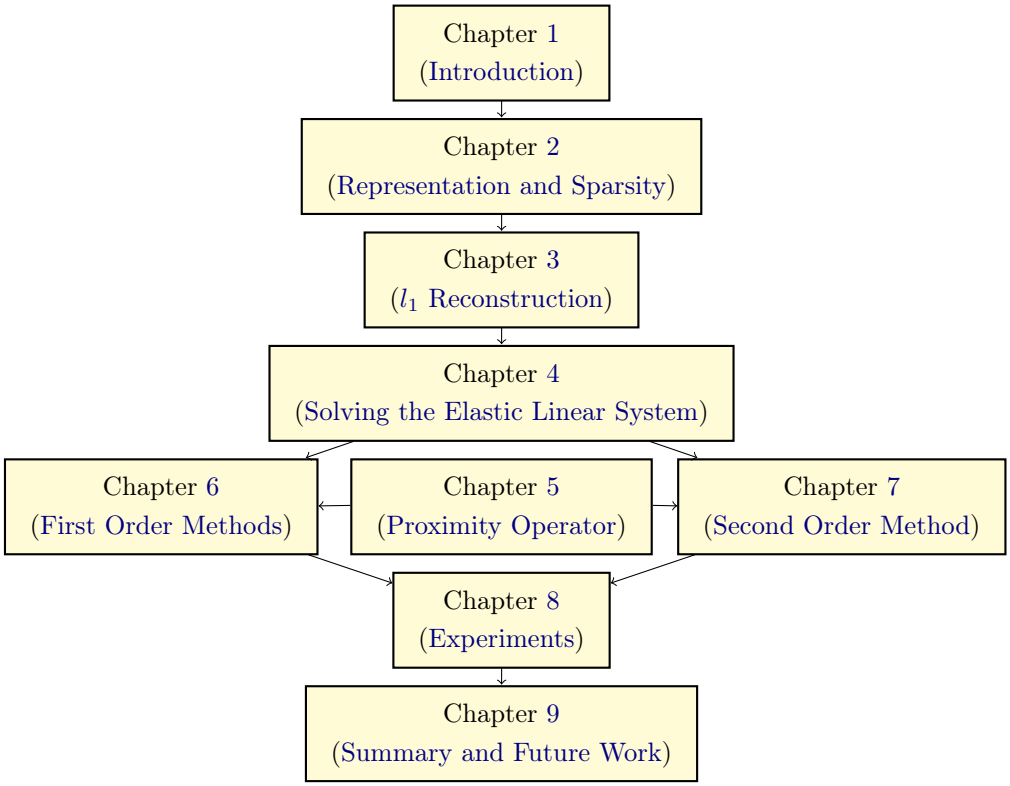



Figure 1.1: Graph of chapter dependency.

Chapter 2

# Representation and Sparsity

## Contents



In this chapter, we explain how images are represented. We also show that images considered in this thesis are sparse in some domains. We do this, by analysing their histograms under some transformations. Thus, although we don't provide any formal definition of the sparsity the reader should acquire some intuitions that lie behind this concept.

Section 2.1 shows the representation of images that is used in this thesis. Briefly, images are stored in the vector-form by taking their columns and gluing them together. Section 2.2 introduces the concept of sparsity and sparsifying transform which can be seen as a generalization of the former. Finally, the concept of compressed sensing is briefly introduced in Section 2.3.

## 2.1 Representation of Images

Ideally, the image would be a continuous mapping $u$ from the $\mathbb{R}^2$ space into some set $\mathcal{A}$ called the co-domain of $u$. In practice, however, the domain of the image $u$ is some discrete and bounded subset $\Omega$ of the set $\mathbb{R}^2$. So in reality we have $u : \Omega \subset \mathbb{R}^2 \to \mathcal{A}$. If we assume that $u(x, y)$ represents the light intensity which was measured at the point $(x, y)$ then we should take $\mathcal{A} := \mathbb{R}$ as the co-domain. Similarly, if we are interested in color RGB images then we should consider the following co-domain $\mathcal{A} := \mathbb{R}^3$. In this thesis, however, we consider complex-valued images, so in our case $\mathcal{A} := \mathbb{C}$.

Since we assumed that the domain of the image is discrete and bounded we can represent images using the matrix notation. Therefore, if $n_x$ and $n_y$ denotes respectively the number of columns and the number of rows of the image $u$, we can assume that $\boldsymbol{u} \in \mathbb{C}^{n_y \times n_x}$ and write $\boldsymbol{u}_{y,x}$ instead of $u(x, y)$. We use notation $\boldsymbol{u}$ to denote the matrix-form or the vector-form representation. Although, this is a convenient notation we rather use different one to denote images. For every image $\boldsymbol{u} \in \mathbb{C}^{n_y \times n_x}$ we can consider its vectorized version $\boldsymbol{g} \in \mathbb{C}^{n_y n_x}$ obtained by taking columns of $\boldsymbol{u}$ and gluing them together to form a vector. Using the matlab-like notation this corresponds to writing $\boldsymbol{g} := reshape(\boldsymbol{u}, n_x n_y, 1)$.

Taking the implementation into account, the reader should remember that by using the vectorized notation, information about the number of rows $n_y$ and the number of columns $n_x$

is lost. So in practice either images should be implemented by using the matrix representation or the information about $n_x$ and $n_y$ should be kept on the side. Additionally, in order to visualize images we transform complex-valued images to the real-valued space by taking absolute values of their coefficients.

## 2.2 Sparsity of Images

The image $\boldsymbol{u}$ is strongly sparse if it can be represented as a linear combination of small number of basis $\boldsymbol{b}_i$, that is $\boldsymbol{u} = \sum_{i \in \mathcal{D}} \alpha_i \boldsymbol{b}_i$ providing that the cardinality of $\mathcal{D}$ is small. In practice images are often weakly sparse which means that most of the coefficients are small but don't have to be exactly zero. In this thesis, by sparsity we mean weak sparsity.

Histograms of sparse images exhibit characteristic peaks (Figure 2.1). Although, the normalized histogram (Figure 2.2b) of the anatomical MRI scan (Figure 2.2a) also contains the characteristic peak, it also contains 'bumps' and so the sparsity of the image in real-valued domain is slightly less accentuated than in the case of angiography.

The sparsifying transform maps the image from the one domain into the another domain where the image is assumed to be sparse. We call this kind of domain the sparse domain. The notion of the sparsifying transform can be seen as a generalization of the sparsity itself, that is if the sparsifying transform is an identity mapping then the mapping gives back the original domain. This concept of generalization is important in practice since although some images are not sparse in the original domain they can exhibit sparsity in some another domain (Figure 2.3). Therefore, if we know the sparsifying transform then we can do the reconstruction in the sparse domain induced by the transform. Figure 2.5 shows the normalized histograms of the image in some sparse domains induced by the imaginary part penalizing operator, the wavelet transform and the finite difference operator. Formal definitions of these operators can be found in Section 3.2, however at this moment the reader, without going into details, can assume that those operators are just some sparsifying transforms.

## 2.3 Compressed Sensing

Compressed sensing is a mathematical framework which deals with the problem of reconstruction of images from the small number of measurements. This approach requires three ingredients:

- The measurement matrix $\boldsymbol{X}$ with $m$ rows and $n$ columns.
- The reconstruction model.
- The sparsity assumption.

Figure 2.4 shows the two major ingredients of the compressed sensing together with the sparsifying transform assumption.

The problem itself can be mathematically modeled as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{u} + \varepsilon$$

where $\boldsymbol{u} \in \mathbb{R}^n$ is the original image, $\boldsymbol{y} \in \mathbb{R}^m$ is the observation and $\varepsilon$ is noise. In addition, since we are interested in taking small number of measurements, we assume that $m << n$. In this thesis we use two different methods of producing measurement matrices, the first one proposed in Lustig et al. [2007] and the second one proposed in Seeger et al. [2009b].

The reconstruction model relies on the sparsity assumption and usually two norms are involved:

- The $l_2$-norm $||\cdot||_{l_2}$ which measures deviations from the observation.
- The $l_1$-norm $||\cdot||_{l_1}$ which can be seen as a sparsity-inducing term.

Therefore if we expect that a solution is sparse in the original domain the following model could be suitable

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2+\kappa||\boldsymbol{u}||_{l_1} \tag{2.1}$$

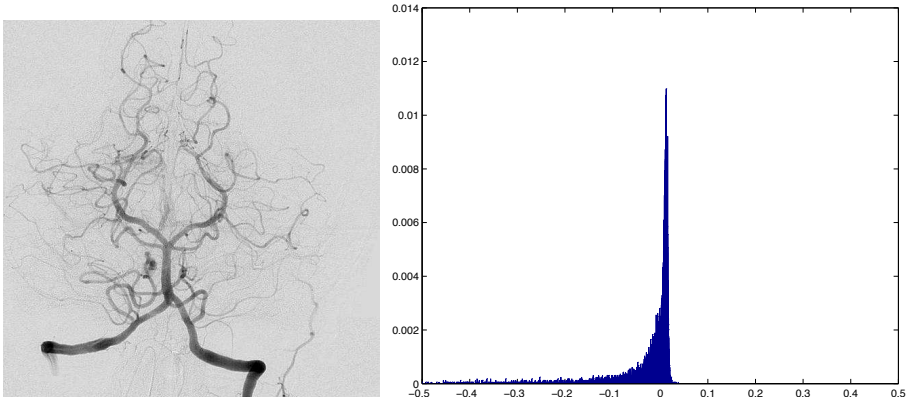
where $\kappa \in \mathbb{R}_+$ is a model parameter and $\mathbb{R}_+$ denotes the set of nonnegative real numbers. On the other hand, if we expect from a solution to be sparse in a wavelet domain then we should consider the following model

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2+\kappa||\boldsymbol{B}_a\boldsymbol{u}||_{l_1} \tag{2.2}$$

where $\boldsymbol{B}_a$ is an orthonormal wavelet transform. More general, if $\boldsymbol{\Psi}$ is a sparsifying transform then the following model can be considered

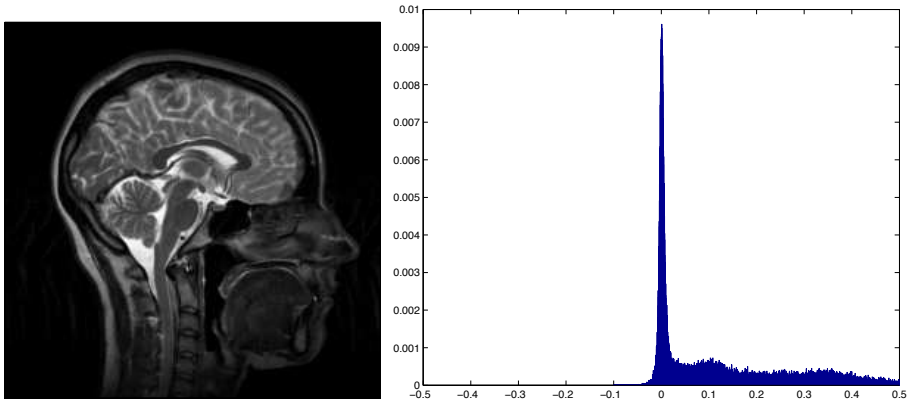
$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min}\ \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2+\kappa||\boldsymbol{\Psi}\boldsymbol{u}||_{l_1} \tag{2.3}$$

The reader can find more about the reconstruction model used in this thesis in Section 3.3.




(a) Angiogram (b) Normalized histogram

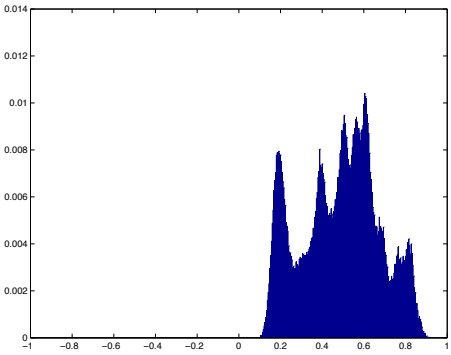
Figure 2.1: Angiogram (Figure 2.1a) showing a transverse projection of the vertebrobasilar and the posterior cerebral circulation (http://en.wikipedia.org/wiki/Angiography). Coefficients of the image were shifted by 0.6. Its normalized histogram (Figure 2.1b) has sharp peak which is also characteristic for weakly sparse images.



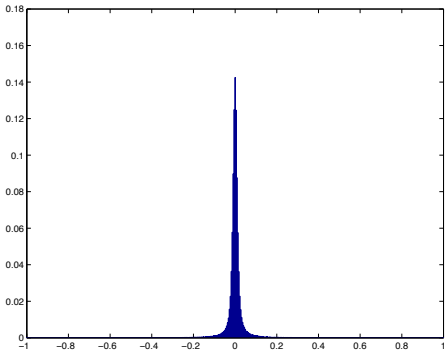

(a) MR image (b) Pixel representation (only the real part)

Figure 2.2: Figure 2.2a shows the sagittal MR image of the subject 1 (slice 8, TE=92ms). Since the image is complex-valued, the absolute value of every coefficient is shown. Figure 2.2b shows the normalized histogram of the Figure 2.2a in the pixel domain (only the real part of the image is shown).

(a) The 'natural' image

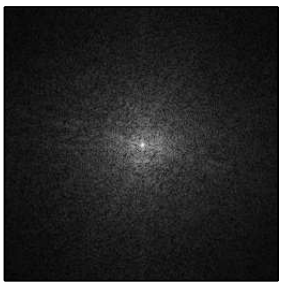



(b) Pixel representation

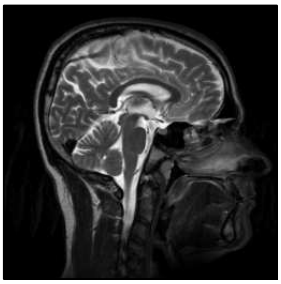



(c) Wavelet representation

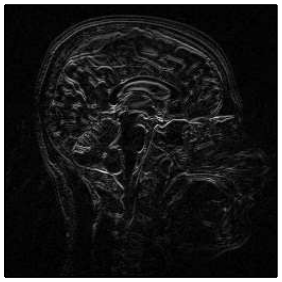
Figure 2.3: Normalized histograms of natural images might not be sparse. Most coefficients of the normalized histogram (Figure 2.3b) are not small enough and there is more than one peak. However, sparsity of the image can be still achieved in some another representation, for example in the wavelet-domain (Figure 2.3c).



*Measurement*

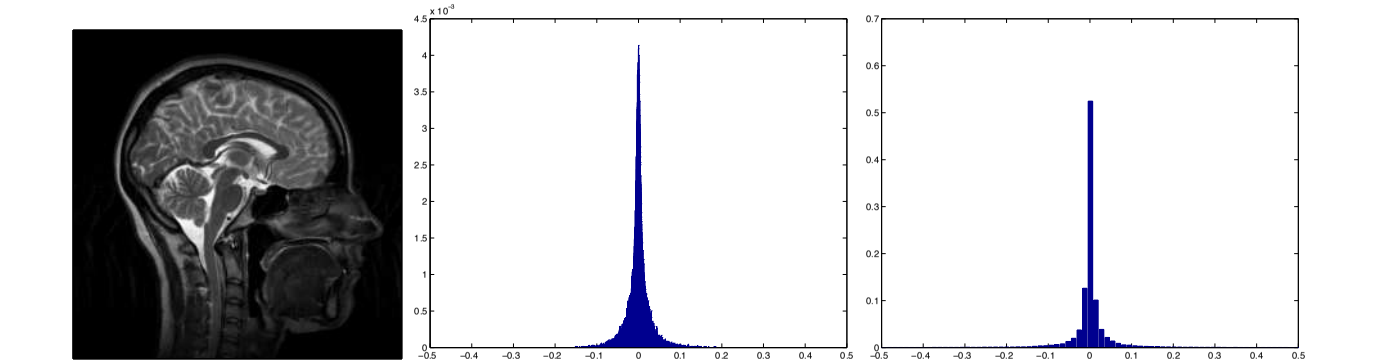


*Reconstruction*



*Sparsity*

Figure 2.4: Two main ingredients of the compressed sensing: a measurement matrix and a reconstruction model. The third ingredient is the assumption that the image itself is sparse in some domain.



(a) MR image (b) Imaginary-valued domain (c) Wavelet domain (the real part)

(d) Wavelet domain (the imaginary part) (e) Finite difference domain (the real part) (f) Finite difference domain (the imaginary part)

Figure 2.5: It is shown normalized histograms of the image (Figure 2.5a; subject 1, slice 8, TE=92ms) in some sparse domains.

Chapter 3

# $l_1$ Reconstruction

## Contents



In this chapter, the reconstruction model that is central to this thesis is introduced. This model itself is formulated as a convex optimization problem and consist of two major terms

- The data fitting term which measures deviations from the observed image. The smaller this term is, the closer the image is to the observation.
- The regularization term which enforces sparsity. The smaller this term is, the sparser the image is.

Section 3.1 shows the real-valued surrogate of the complex field $\mathbb{C}$. This notation is useful whenever work with real-valued models instead of the corresponding complex-valued models is wanted. Section 3.2 introduces operators used in the reconstruction model. Although, they are presented in the matrix-form, they are not stored in this form. Instead, a function application whenever the matrix-vector multiplication takes place is performed. Section 3.3 introduces the MRI energy function which is the objective function of the optimization problem. Section 3.4 shows how the transposition of the operators should be interpreted. In this thesis we have to deal with operators that are defined in both real-valued space and complex-valued space, and in order to make calculations easier proper interpretation of the transposition is needed.

The following notation $\boldsymbol{u}_{(j)}$ is used to distinguish between different vectors and components of the vector. For example, $\boldsymbol{u}_{(a)}$ and $\boldsymbol{u}_{(r)}$ denote different vectors, whereas $\boldsymbol{u}_j$ denotes $j$-th entry of the vector $\boldsymbol{u} \in \mathcal{C}^n$. However, we use notation $\boldsymbol{B}_a$ and $\boldsymbol{B}_r$ to denote different

operators. In the same way we distinguish scalars, that is $\mu_a$ and $\mu_r$ are different scalars. If we want to refer to a particular entry of some matrix $\boldsymbol{A}$ then we use notation with double indices, that is $\boldsymbol{A}_{k,l}$ denotes the $(k,l)$ entry of the matrix $\boldsymbol{A}$.

The notation $\boldsymbol{u}^{\mathcal{M}}$ is used to denote matrix-form representation of the vector $\boldsymbol{u}$. Precise definitions can be found in Appendix B.

## 3.1 Real-valued Surrogate of the Complex Field

Put $\mathcal{C} := \mathbb{R}^2$ to be the real-valued surrogate of the complex field $\mathbb{C}$ and assume from now that calculations are done in $\mathcal{C}^n$ instead of $\mathbb{C}$. If $\boldsymbol{v} \in \mathbb{R}^n$ then $v_i \in \mathbb{R}$ denotes the i-th entry of the vector $\boldsymbol{v}$. Similarly, if $\boldsymbol{u} \in \mathcal{C}^n$ then

$$\boldsymbol{u}_i = \begin{bmatrix} \Re u_i \\ \Im u_i \end{bmatrix}$$

denotes the i-th entry of the vector $\boldsymbol{u}$ and $\boldsymbol{u}_i \in \mathcal{C}$. Under this definition the absolute value of $z \in \mathbb{C}$ can be seen as

$$|\Re z + i\, \Im z| = \left|\left| \begin{bmatrix} \Re z \\ \Im z \end{bmatrix} \right|\right|_{l_2}$$

and $l_1$-norm can be stated as a sum of $l_2$ norms. More precisely, if $\boldsymbol{u} \in \mathcal{C}^n$ then

$$||\boldsymbol{u}||_{l_1} = \sum_j ||\boldsymbol{u}_j||_{l_2}$$

Both sets $\mathcal{C}^n$ and $\mathbb{R}^{2n}$ are transparent and can be used interchangeably.

## 3.2 Operators

The reconstruction model that we consider depends on the four matrices (also called operators and some of them - transforms):

- The measurement matrix $\boldsymbol{X}$.
- The wavelet transform $\boldsymbol{B}_a$.
- The finite difference operator $\boldsymbol{B}_r$.
- The operator $\boldsymbol{B}_i$ that penalizes the existence of the imaginary part.

The measurement matrix $\boldsymbol{X}$ is a part of the data fitting term, whereas other operators occur in the regularization term.

### 3.2.1 Discrete Fourier Operator

Consider a complex-valued vector $\boldsymbol{u} \in \mathbb{C}^n$ and its matrix form $\boldsymbol{u}^{\mathcal{M}}$. Moreover, let the matrix $\boldsymbol{u}^{\mathcal{M}}$ has $n_y$ rows and $n_x$ columns, and $n = n_x \cdot n_y$. The 2-D discrete Fourier transform $\boldsymbol{F}^{\mathcal{M}}$

is an operator defined as follows

$$(\boldsymbol{F}\boldsymbol{u})^{\mathcal{M}}_{\gamma,\omega} := \frac{1}{\sqrt{n}} \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} \boldsymbol{u}^{\mathcal{M}}_{y,x} \exp\left(-i2\pi\left(\frac{x\cdot\omega}{n_x} + \frac{y\cdot\gamma}{n_y}\right)\right) \tag{3.1}$$

In this thesis we use $\boldsymbol{F}$ to denote the discrete Fourier operator where the operation $\boldsymbol{F}\boldsymbol{u}$ can be described as follows:

- Convert the vector $\boldsymbol{u}$ into its matrix-form $\boldsymbol{u}^{\mathcal{M}}$.
- Apply the 2-D discrete Fourier transform to the matrix $\boldsymbol{u}^{\mathcal{M}}$. Let the matrix $\overline{\boldsymbol{u}}^{\mathcal{M}}$ be the result of this application.
- Convert the matrix $\overline{\boldsymbol{u}}^{\mathcal{M}}$ into its vector-form $\overline{\boldsymbol{u}}$.

Conversion can be done by using matlab-like $reshape(\boldsymbol{A}, m, n)$ operation[1] which returns the $m$-by-$n$ matrix $\boldsymbol{B}$. The elements of the matrix $\boldsymbol{B}$ were taken column-wise from the matrix $\boldsymbol{A}$. Then the conversion from the matrix-form $\boldsymbol{u}^{\mathcal{M}}$ into the vector-form $\boldsymbol{u}$ can be defined as $\boldsymbol{u} := reshape(\boldsymbol{u}^{\mathcal{M}}, n, 1)$. Similarly, the conversion from the vector-form $\boldsymbol{u}$ into the matrix-form $\boldsymbol{u}^{\mathcal{M}}$ can be defined as $\boldsymbol{u}^{\mathcal{M}} := reshape(\boldsymbol{u}, n_y, n_x)$.

Apart from the discrete Fourier transform we also need its inverse. Define

$$(\boldsymbol{F}^H\ \boldsymbol{u})^{\mathcal{M}}_{y,x} := \frac{1}{\sqrt{n}} \sum_{\omega=0}^{n_x-1} \sum_{\gamma=0}^{n_y-1} \boldsymbol{u}^{\mathcal{M}}_{\gamma,\omega} \exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x} + \frac{y\cdot\gamma}{n_y}\right)\right) \tag{3.2}$$

We can define $\boldsymbol{F}^H$ by using the same procedure to the one that we exploited in order to establish $\boldsymbol{F}$.

It can also be shown that $(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}} = (\boldsymbol{F}^H(\boldsymbol{F}\boldsymbol{u}))^{\mathcal{M}} = \boldsymbol{u}^{\mathcal{M}}$ (see Theorem D.1) which shows that $\boldsymbol{F}^H$ is really the inverse of $\boldsymbol{F}$.

### 3.2.2 Subsampling Fourier Operator

Consider the measurement matrix. In our setting, this matrix consists of two parts:

- The discrete Fourier transform which appears in MRI setting due to sampling of the k-space.
- The subsampling operator which cuts off some columns in the Fourier spectrum.

Intuitively, the latter operator transforms an image from the high resolution one to the low resolution one. We start from this operator

**Definition 3.1** (Subsampling Operator)**.** *Let $\mathcal{C}$ be the real-valued surrogate of the complex field $\mathbb{C}$ and $\boldsymbol{v} \in \mathcal{C}^n$. For fixed subset of indices $\mathcal{J} \subset \{0, 1, .., n-1\}$ let define*

$$\boldsymbol{I}_{\mathcal{J},\cdot}\ \boldsymbol{v} := \Big[\boldsymbol{v}_j\Big]_{j\in\mathcal{J}}$$

*The operator $\boldsymbol{I}_{\mathcal{J},\cdot}$ is the subsampling operator and set $\mathcal{J}$ is the set of chosen indices.*

[1] MathWorks: http://www.mathworks.com/help/techdoc/ref/reshape.html.

and its adjoint

**Definition 3.2** (Upsampling Operator)**.** *Let $\mathcal{C}$ be the real-valued surrogate of the complex field $\mathbb{C}$ and $\boldsymbol{v} \in \mathcal{C}^m$. For fixed subset of indices $\mathcal{J} \subset \{0, 1, .., n-1\}$ let define*

$$(\boldsymbol{I}_{\cdot,\mathcal{J}}\ \boldsymbol{v})_j := \begin{cases} \boldsymbol{v}_j & \text{if } j \in \mathcal{J} \\ \boldsymbol{0} & \text{otherwise} \end{cases}$$

*where $\boldsymbol{0} \in \mathbb{R}^2$ is the zero vector. The operator $\boldsymbol{I}_{\cdot,\mathcal{J}}$ is the upsampling operator.*

Equipped with the subsampling operator we can define the subsampling Fourier operator $\boldsymbol{X}$. Note that this matrix depends on a set of indices that has to be chosen in advance. Different choices of this set lead to different subsampling schemes, and consequently to different reconstructions. So the reconstruction may be poorer under some choices of this set. Often, the smaller the set, the harder the reconstruction. Therefore small sets of the chosen indices usually lead to poor reconstructions.

**Definition 3.3** (Subsampling Fourier Operator)**.** *Put*

$$\boldsymbol{X} := \boldsymbol{I}_{\mathcal{J},\cdot}\boldsymbol{F}$$

*where $\boldsymbol{F}$ is the discrete Fourier operator and $\boldsymbol{I}_{\mathcal{J},\cdot}$ is the subsampling operator for fixed set of chosen indices $\mathcal{J} \subset \{0, 1, .., n-1\}$.*

Similarly, we can define the upsampling inverse Fourier operator

**Definition 3.4** (Upsampling Inverse Fourier Operator)**.** *Put*

$$\boldsymbol{X}^H := \boldsymbol{F}^H\boldsymbol{I}_{\cdot,\mathcal{J},\cdot}$$

*where $\boldsymbol{F}^H$ is the discrete inverse Fourier operator and $\boldsymbol{I}_{\cdot,\mathcal{J},\cdot}$ is the upsampling operator for fixed set of chosen indices $\mathcal{J} \subset \{0, 1, .., n-1\}$.*

Note that $\boldsymbol{X}$ contains orthonormal rows and $\boldsymbol{X}\boldsymbol{X}^H = \boldsymbol{I}$ holds but it doesn't have to be $\boldsymbol{X}^H\boldsymbol{X} = \boldsymbol{I}$. In this thesis the subsampling operator can be identified with dropping columns of the matrix $\boldsymbol{v}^{\mathcal{M}}$.

### 3.2.3 Wavelet Operator

We used an orthonormal wavelet transform (Daubechies wavelet) to perform operation $\boldsymbol{B}_a\boldsymbol{u}$ where $\boldsymbol{B}_a$ denotes the wavelet operator.

The wavelet operator will not be explained in this thesis. For further reading please see Mallat [2008].

### 3.2.4 Finite Difference Operator

The finite difference operator contains two 'building blocks': the horizontal finite difference operator and the vertical finite difference operator. So, in order to define the finite difference operator, first we should have a look at these two operators.

The horizontal finite difference operator is a discrete approximation of the continuous differential operator $\partial$ along the 'horizontal direction'. Definition 3.6 introduces the horizontal finite difference operator and relates the operator to the swapping matrix. The latter is defined as follows

**Definition 3.5** (Swapping Matrix)**.** *Let* $\boldsymbol{A}_{(n)} \in \mathbb{R}^{n \times n}$ *be*

$$\boldsymbol{A}_{(n)} = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

*The matrix* $\boldsymbol{A}_{(n)}$ *is called the swapping matrix.*

Now we define the horizontal finite difference operator

**Definition 3.6** (Horizontal Finite Difference Operator)**.** *Let* $\boldsymbol{P}_h$ *be a permutation operator such that*

$$\boldsymbol{P}_h = \begin{pmatrix} \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{I}_{n_y} \\ \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \end{pmatrix} = \boldsymbol{A}_{(n_x)} \otimes \boldsymbol{I}_{n_y}$$

*where* $\boldsymbol{0}^{n_y}$ *denotes the* $n_y$*-by-*$n_y$ *block-matrix of zeros,* $\boldsymbol{I}_{n_y}$ *denotes the* $n_y$*-by-*$n_y$ *identity matrix and* $\boldsymbol{A}_{(n_x)}$ *is the swapping matrix. The operator defined as*

$$\nabla_{1,h} := \boldsymbol{P}_h - \boldsymbol{I}$$

*is the horizontal finite difference operator.*

Note that for a given $\boldsymbol{u}$ we have $(\boldsymbol{P}_h \boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{(y,[x+1]_{n_x})}$ where $[\cdot]_{n_x}$ is the cycling operator [2]. Therefore we also have $(\nabla_{1,h}\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{(y,[x+1]_{n_x})} - \boldsymbol{u}^{\mathcal{M}}_{(y,x)}$.

Now we can easily derive transpose of the horizontal operator $\nabla^T_{1,h}$

$$\nabla^T_{1,h} = \boldsymbol{P}_h^T - \boldsymbol{I}$$

where

$$\boldsymbol{P}_h^T = \begin{pmatrix} \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} \\ \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{I}_{n_y} & \dots & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{I}_{n_y} & \boldsymbol{0}^{n_y} \end{pmatrix} = \boldsymbol{A}^T_{(n_x)} \otimes \boldsymbol{I}_{n_y}$$

[2] $[a]_{n_x} := a \mod n_x$

So $\boldsymbol{P}_h^T$ is also a permutation operator $(\boldsymbol{P}_h^T\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{(y,[x-1]_{n_x})}$ and $(\nabla_{1,h}^T\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{(y,[x-1]_{n_x})} - \boldsymbol{u}^{\mathcal{M}}_{(y,x)}$.

The vertical finite difference operator is the second operator which is required to fully define the finite difference operator. It approximates continuous differential operator $\partial$ along the 'vertical direction'.

**Definition 3.7** (Vertical Finite Difference Operator)**.** *Let*

$$\boldsymbol{P}_v = \begin{pmatrix} \boldsymbol{A}_{(n_y)} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{A}_{(n_y)} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{A}_{(n_y)} & \dots & \boldsymbol{0}^{n_y} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{A}_{(n_y)} \end{pmatrix} = \boldsymbol{I}_{n_x} \otimes \boldsymbol{A}_{(n_y)}$$

*where $\boldsymbol{0}^{n_y}$ is the $n_y$-by-$n_y$ block-matrix of zeros, $\boldsymbol{I}_{n_x}$ denotes the $n_x$-by-$n_x$ identity matrix and $\boldsymbol{A}_{(n_y)}$ is the swapping matrix. The operator defined as*

$$\nabla_{1,v} := \boldsymbol{P}_v - \boldsymbol{I}$$

*is the vertical finite difference operator.*

Note that for a given $\boldsymbol{u}$ we have $(\boldsymbol{P}_v\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{([y+1]_{n_y},x)}$ and so $(\nabla_{1,v}\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{([y+1]_{n_y},x)} - \boldsymbol{u}^{\mathcal{M}}_{(y,x)}$. The transpose of $\boldsymbol{P}_v$ is

$$\boldsymbol{P}_v^T = \begin{pmatrix} \boldsymbol{A}_{(n_y)}^T & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{A}_{(n_y)}^T & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{0}^{n_y} \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{A}_{(n_y)}^T & \dots & \boldsymbol{0}^{n_y} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \boldsymbol{0}^{n_y} & \dots & \boldsymbol{A}_{(n_y)}^T \end{pmatrix} = \boldsymbol{I}_{n_x} \otimes \boldsymbol{A}_{(n_y)}^T$$

So $\boldsymbol{P}_v^T$ is also a permutation matrix and $(\boldsymbol{P}_v^T\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{([y-1]_{n_y},x)}$ holds. Since $\nabla_{1,v}^T = \boldsymbol{P}_v^T - \boldsymbol{I}$, we have $(\nabla_{1,v}^T\boldsymbol{u})^{\mathcal{M}}_{(y,x)} = \boldsymbol{u}^{\mathcal{M}}_{([y-1]_{n_y},x)} - \boldsymbol{u}^{\mathcal{M}}$.

Finally, we can define the finite difference operator

**Definition 3.8** (Finite Difference Operator)**.** *The whole finite difference operator can be expressed as*

$$\nabla_1 := \begin{bmatrix} \nabla_{1,h} \\ \nabla_{1,v} \end{bmatrix}$$

*We also use symbol $\boldsymbol{B}_r$ to denote the finite difference operator, that is $\boldsymbol{B}_r := \nabla_1$.*

### 3.2.5 Total Variation Seminorm

The finite difference operator can be used to define seminorm. This leads to the so called total variation seminorm. In this thesis the following definition is used

**Definition 3.9** (Anisotropic Total Variation Seminorm)**.** *Let $\nabla_1$ be a finite difference operator. The (anisotropic) total variation seminorm for $\boldsymbol{u}$ is*

$$||\boldsymbol{u}||_{TV} := ||\nabla_1 \boldsymbol{u}||_{l_1}$$

*We may use the shorter name TV-seminorm to denote the (anisotropic) total variation seminorm.*

The total variation seminorm on the one hand induces smoothness of the image and on the other hand it preserves sharp edges. This regularization term was proposed in Rudin et al. [1992] in order to denoise images.

The finite difference operator $\boldsymbol{B}_r$ derived in Subsection 3.2.4 leads to the circular and anisotropic version of the total variation seminorm.

### 3.2.6 Imaginary Part Penalizing Operator

In the context of the MR images' reconstruction, it is useful to penalize the imaginary part (the imaginary part of the MR image might not vanish due to acquisition errors). This can be done by introducing special operator

**Definition 3.10** (Imaginary Part Penalizing Operator)**.** *Let $\mathcal{C} := \mathbb{R}^2$ be the real-valued surrogate of the complex field and $\boldsymbol{u} \in \mathcal{C}^n$ be such that $\boldsymbol{u}_j := [\Re u_j, \Im u_j]^T \in \mathcal{C}$. Let $\boldsymbol{B}_i \in \{0,1\}^{n \times 2n}$ be defined as*

$$\boldsymbol{B}_i \boldsymbol{u} := [\Im u_j]_{j \in \{0,1,\ldots,n-1\}}$$

*Such operator $\boldsymbol{B}_i$ is the imaginary part penalizing operator.*

Let look more precisely how matrix $\boldsymbol{B}_i$ can be defined

$$\boldsymbol{B}_i = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} = \boldsymbol{I} \otimes \boldsymbol{\delta}_2^T$$

So its transpose is

$$\boldsymbol{B}_i^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} = \boldsymbol{I} \otimes \boldsymbol{\delta}_2$$

where $\boldsymbol{\delta}_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. It is easy to see that $\boldsymbol{B}_i$ drops the 'real positions' and $\boldsymbol{B}_i^T$ fills vector with zeros at the 'real positions'. More precisely, if $\boldsymbol{v} \in \mathbb{R}^n$ then $\boldsymbol{B}_i^T \in \{0,1\}^{2n \times n}$ and

$$\forall_{\boldsymbol{v} \in \mathbb{R}^n} \; \boldsymbol{B}_i^T \boldsymbol{v} = \begin{bmatrix} 0 \\ v_j \end{bmatrix}_{j \in \{0,1,\ldots,n-1\}}$$

Moreover $\boldsymbol{B}_i \boldsymbol{B}_i^T \in \{0,1\}^{n \times n}$, $\boldsymbol{B}_i^T \boldsymbol{B}_i \in \{0,1\}^{2n \times 2n}$,

$$\boldsymbol{B}_i \boldsymbol{B}_i^T = \boldsymbol{I}$$

and

$$\boldsymbol{B}_i^T \boldsymbol{B}_i = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} = \boldsymbol{I} \otimes \boldsymbol{\delta}_2 \boldsymbol{\delta}_2^T$$

Notice also the subtle difference between the following operators:

1. $\boldsymbol{A}\boldsymbol{v} := \begin{bmatrix} 0 \\ v_j \end{bmatrix}_j$
2. $\boldsymbol{C}\boldsymbol{v} := [v_j]_j$

For every $\boldsymbol{v} \in \mathcal{C}^n$ the first operator returns a vector which is still in $\mathcal{C}^n$ space (but with 0's at the 'real positions'), whereas the second operator maps from $\mathcal{C}^n$ to $\mathbb{R}^n$.

### 3.2.7 Implementation Issues

Although, the matrix-form representation of the operators was shown, in practice, it would not be possible to store all operators in this form. For example, dealing with the real-valued image of the size $256 \times 256$ leads to the matrix-form representation of some operators to have $256^4 = 4294967296$ entries. This is far too much if we wanted to store them explicitly. Fortunately, there is no need to store operators in this form. Instead, they were implemented as functions that can be applied to vectors. For instance, implementation of the fast Fourier transform (FFT)[3] was used to represent the operation $\boldsymbol{F}\boldsymbol{u}$ and the inverse fast Fourier transform[4] was used to represent the operation $\boldsymbol{F}^H\boldsymbol{u}$; provided that the set of chosen indices $\mathcal{J}$ was given the subsampling operator $\boldsymbol{I}_{\mathcal{J},\cdot}$ uses matlab-like instruction $\boldsymbol{u}^{\mathcal{M}}(:,\mathcal{J})$ to implement $\boldsymbol{I}_{\mathcal{J},\cdot}\boldsymbol{u}$; the fast implementation of the wavelet transform[5] was used to represent

[3] MathWorks: http://www.mathworks.com/help/techdoc/ref/fft2.html; one should also check if the current implementation corresponds to the orthonormal discrete Fourier transform.

[4] MathWorks: http://www.mathworks.com/help/techdoc/ref/ifft2.html.

[5] FWTN written by Hannes Nickisch: http://hannes.nickisch.org/code/index.html.

operation $\boldsymbol{B}_a\boldsymbol{u}$; matlab function *diff* was executed whenever the finite difference operator $\boldsymbol{B}_r$ was used; and finally, matlab-like expression $\boldsymbol{u}(2:2:end)$ was executed whenever the operation $\boldsymbol{B}_i\boldsymbol{u}$ took place.

Matrix-vector multiplication operations of the operators exhibit the following time complexity $c(\boldsymbol{X}\boldsymbol{v}) = O(n\log(n))$ and $\forall_{j\in\{a,r,i\}}\ c(\boldsymbol{B}_j\boldsymbol{v}) = O(n)$ where $n$ accounts for the number of entries of the vector $\boldsymbol{v}$ and $c(\boldsymbol{A}\boldsymbol{v})$ denotes the time complexity of performing the operation $\boldsymbol{A}\boldsymbol{v}$.

## 3.3 $l_1$ Reconstruction of the MR Images

Consider the reconstruction of a MR image from the incomplete Fourier measurements under the assumption that the image has sparse representation under some transformation. Let $\boldsymbol{u} \in \mathbb{C}^n$ be the complex-valued image. Suppose that $\boldsymbol{y} \in \mathbb{C}^m$ is the noisy observation given by the subsampling Fourier operator $\boldsymbol{X}$. The relation between the observation $\boldsymbol{y}$ and the image $\boldsymbol{u}$ is modeled as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{u} + \varepsilon \tag{3.3}$$

where $\varepsilon \sim N(0, \sigma^2\boldsymbol{I})$ is interpreted as noise. In this thesis the matrix $\boldsymbol{X}$ is called the measurement matrix. Let $\boldsymbol{\Psi}$ be a sparsifying transform; the linear operator that transforms the image from the pixel-valued representation to the sparse representation. The reconstruction problem can now be formulated as the following unconstrained optimization problem

$$\arg\min_{\boldsymbol{u}\in\mathbb{C}^n} \quad \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \kappa||\boldsymbol{\Psi}\boldsymbol{u}||_{l_1} \tag{3.4}$$

Intuitively, formula (3.4) can be interpreted as a trade-off between the fidelity of the reconstruction to the measured data where the error is measured as a squared $l_2$ norm $||\cdot||_{l_2}^2$ and sparsity of the vector $\boldsymbol{\Psi}\boldsymbol{u}$. It is well-known that the latter can be done by the sparsity-inducing term $||\cdot||_{l_1}$ [Chen et al. 1999; Lustig et al. 2007; Seeger et al. 2009b]. The parameter $\kappa$ can be seen as a trade-off parameter. So, in other words, we want to find a solution $\boldsymbol{u}$ that is both 'close' to the observation $\boldsymbol{y}$ and $\boldsymbol{\Psi}\boldsymbol{u}$ is 'sparse enough'.

The objective function of the optimization problem (3.4) is not differentiable. Therefore some popular methods such as *Non-linear Conjugate Gradient* [Lustig et al. 2007] cannot be directly applied. One may consider the differentiable surrogate of the $||\cdot||_{l_1}$ defined as

$$||\boldsymbol{z}||_\varepsilon := \sum_j \sqrt{|z_j|^2 + \varepsilon} \text{ for every } \boldsymbol{z} \in \mathbb{C}^n$$

and convert the non-differentiable objective function into the differentiable one.

In the MRI setting it is useful to optimize over the complex-valued domain instead of the real-valued one. The reason lies on the imperfections of the acquisition process, that is because of resonance frequency offsets, magnetic field inhomogeneities or eddy currents, the reconstruction contains a phase [Bernstein et al. 2004; Seeger et al. 2009a].

Since some expressions in problem (3.4) are not complex analytic we cannot work directly in the complex vector space. Instead we use $\mathbb{R}^2$ as a surrogate of the complex field $\mathbb{C}$ (Section

3.1). This leads to the real-valued models over twice as many variables as the corresponding complex-valued models contain.

In the MRI setting we may consider each of the following operators, the wavelet transform $\boldsymbol{B}_a$, the finite difference operator $\boldsymbol{B}_r$ and the imaginary part penalizing operator $\boldsymbol{B}_i$ to be the sparsifying transform. Although the imaginary part penalizing operator works already with the real-valued surrogate of the complex set, we have to slightly redefine the other operators:

$$\forall_{j\in\{a,r\}}\ \boldsymbol{B}_j := \boldsymbol{B}_j \otimes \boldsymbol{I}_2 \tag{3.5}$$

The optimization problem suitable for the reconstruction of MR images uses the following objective function

**Definition 3.11** (MRI Energy Function)**.** *Let $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$, $\boldsymbol{X}$ be, respectively, the wavelet transform, the finite difference operator, the imaginary part penalizing operator and the subsampling Fourier operator, and assume that all of them are tailored to work with the real-valued surrogate of the complex set $\mathcal{C} := \mathbb{R}^2$. For every $\boldsymbol{u} \in \mathcal{C}^n$ let*

$$\Psi^{MRI}(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_{l_1} + \kappa_r||\boldsymbol{B}_r\boldsymbol{u}||_{l_1} + \kappa_i||\boldsymbol{B}_i\boldsymbol{u}||_{l_1} \tag{3.6}$$

*where for each $j \in \{a, r, i\}$ $\kappa_j := \tau_j\sigma$, $\tau_j$ is a model parameter and $\sigma^2$ is the noise variance. The function $\Psi^{MRI}(\cdot)$ is called the MRI energy function. Operators $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$ are also called penalizing operators.*

Since the subsampling Fourier operator can be written as $\boldsymbol{X} = \boldsymbol{I}_{\mathcal{J},\cdot}\boldsymbol{F}$ (Definition 3.3), different choice of $\boldsymbol{I}_{\mathcal{J},\cdot}$ leads to different measurement matrices. In this thesis two kinds of the measurement matrices are considered. The first one produced by Variable density phase encoding and the second one produced by Bayesian experimental design. These measurement matrices differ from each other by $\boldsymbol{I}_{\mathcal{J},\cdot}$.

Equipped with the MRI energy function we can consider the following model, which is formulated over $\mathcal{C}^n$-space, for reconstructing MR images

$$\underset{\boldsymbol{u}\in\mathcal{C}^n}{\arg\min}\ \Psi^{\mathrm{MRI}}(\boldsymbol{u}) \tag{3.7}$$

By putting more than one sparsifying transform, we expect that the reconstruction is sparse in more than one transform-domain[6]. It is known that 'natural' images are sparse under the wavelet transform [Taubman and Marcellin 2002]. The study of MR images have shown that the orthonormal wavelet transform or the finite difference operator can be also used as a sparsifying transform for MRI (Figure 3 in Lustig et al. [2007] shows the transform-domain sparsity of the axial brain image and the contrast enhanced angiogram of the peripheral leg under the wavelet transform, dct and finite difference operator).

## 3.4 Transpose Operation

In this thesis we often mix operators such as the subsampling Fourier operator $\boldsymbol{X}$ which works with the complex-valued space with operators that work with the real-valued space.

[6] Domain of the transformed image.

Model (3.7) is such an example of mixing operators working in different spaces. Therefore we need clear notion of the transpose operation $(\cdot)^T$.

Let start from the following term

$$||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2$$

where $\boldsymbol{u}$, $\boldsymbol{y}$ and $\boldsymbol{X}$ are complex-valued. From the definition of the norm in the complex-valued space, that is $||\boldsymbol{u}||_{l_2} = \sqrt{\boldsymbol{u}^H\boldsymbol{u}}$, we can derive

$$||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 = ||\boldsymbol{X}\boldsymbol{u}||_{l_2}^2 + ||\boldsymbol{y}||_{l_2}^2 - 2\Re\left((\boldsymbol{X}^H\boldsymbol{y})^H\boldsymbol{u}\right)$$

where $\Re(\boldsymbol{u}) = \boldsymbol{v}$ denotes the vector $\boldsymbol{v}$ obtained from the vector $\boldsymbol{u}$ by taking the real-valued part of the latter. Let $\nu(\cdot) : \mathbb{C} \to \mathcal{C}$ be a mapping from the complex-valued field into the real-valued surrogate defined as

$$\forall_{z:=a+ib\in\mathbb{C}}\ \nu(z) := \begin{bmatrix} a \\ b \end{bmatrix}$$

The extension of this mapping to vectors can readily be obtained by

$$\forall_{\boldsymbol{z}\in\mathbb{C}^n}\ \nu(\boldsymbol{z}) := [\nu(z_j)]_{j=1}^n$$

Note that under this definition of the mapping the term $\Re\left((\boldsymbol{X}^H\boldsymbol{y})^H\boldsymbol{u}\right)$ can be rewritten as $\nu(\boldsymbol{X}^H\boldsymbol{y})^T\nu(\boldsymbol{u})$. Note that the latter is a real-valued dot product. Therefore

$$||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 = ||\nu(\boldsymbol{X}\boldsymbol{u})||_{l_2}^2 + ||\nu(\boldsymbol{y})||_{l_2}^2 - 2\nu(\boldsymbol{X}^H\boldsymbol{y})^T\nu(\boldsymbol{u}) \tag{3.8}$$

and so the right hand side of eq. (3.8) can be computed by using real-valued vectors twice as long as their complex-valued counterparts. Consider now $||\boldsymbol{X}\boldsymbol{u}||_{l_2}^2$, then we have

$$||\boldsymbol{X}\boldsymbol{u}||_{l_2}^2 = ||\nu(\boldsymbol{X}\boldsymbol{u})||_{l_2}^2 = \nu(\boldsymbol{u})^T\boldsymbol{M}\nu(\boldsymbol{u})$$

where $\boldsymbol{M}$ is some real-valued symmetric matrix called here $\boldsymbol{M} = \boldsymbol{X}^H\boldsymbol{X}$. Put $\boldsymbol{A} := \boldsymbol{X}^H\boldsymbol{X} + \sum_{j\in\{a,r,i\}}\mu_j\boldsymbol{B}_j^T\boldsymbol{B}_j$ where $\boldsymbol{B}_a$, $\boldsymbol{B}_r$ and $\boldsymbol{B}_i$ are operators described in Section 3.2. Note that this matrix is symmetric since $\boldsymbol{X}^H\boldsymbol{X}$ and $\boldsymbol{B}_j^T\boldsymbol{B}_j$ are symmetric for every $j \in \{a, r, i\}$.

In this thesis, for the sake of brevity, the mapping $\nu$ is dropped and we use only the real-valued calculations.

Chapter 4

# Solving the Elastic Linear System

## Contents



All algorithms presented in Chapter 7 and some algorithms presented in Chapter 6 require some specific linear system to be solved. In this thesis this system is called the elastic linear system and it can be solved if we concentrate only to the penalizing operators presented in Section 3.2. Moreover, since the dimensionality of the elastic linear system discussed in this chapter is high[1], solving this system should not only be possible but also computationally efficient. In practice, computing the solution of this system should be at most linearithmic[2] with respect to its dimensionality.

Section 4.1 presents the elastic linear system, a system of linear equations to be solved, and which occurs in some algorithms as a subproblem. If operators that occur in the elastic linear system happened to be exactly those defined in Section 3.2 then the whole linear system can be solved. Section 4.2 shows how to do this efficiently. Although in this thesis we mainly work with the three penalizing operators, the wavelet operator, the finite difference operator and the imaginary part penalizing operator, one could consider different set of the penalizing operators. Section 4.2 briefly discusses the possibility of solving the elastic linear system where different penalizing operators are used.

## 4.1 Elastic Linear System

Some algorithms such as *Augmented Lagrangian* (Section 6.3) or *Normed Constrained Quadratic Fast Gradient Projection* (Section 7.3) requires the following problem to be solved

**Definition 4.1** (Elastic Model)**.** *Let* $\boldsymbol{X}$, $\boldsymbol{B}_j$ *be some operators,* $\boldsymbol{u}$, $\boldsymbol{d}_{(j)}$, $\boldsymbol{v}$ *be some vectors and* $\mu_j$ *some parameters for* $j \in \{1, 2, \ldots, K\}$*. The following model*

$$\arg\min_{\boldsymbol{u}} \left\{ E(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \sum_{j=1}^{K} \frac{\mu_j}{2}||\boldsymbol{B}_j\boldsymbol{u} - \boldsymbol{d}_{(j)}||_{l_2}^2 + \boldsymbol{v}^T\boldsymbol{u} \right\} \tag{4.1}$$

[1] If the linear system is $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$, then the dimensionality of this system is equal to the number of entries of the vector $\boldsymbol{u}$.

[2] The time and space complexity of solving the system should be at most $O(n \log(n))$.

*is called the elastic model.*

Taking gradient of the $E(\boldsymbol{u})$ and setting it to zero yields the following linear system to solve

**Definition 4.2** (Elastic Linear System)**.**

$$(\boldsymbol{X}^H\boldsymbol{X} + \sum_{j=1}^{K} \mu_j \boldsymbol{B}_j^T \boldsymbol{B}_j)\boldsymbol{u} = \boldsymbol{X}^H\boldsymbol{y} + \sum_{j=1}^{K} \mu_j \boldsymbol{B}_j^T \boldsymbol{d}_{(j)} - \boldsymbol{v} \tag{4.2}$$

*which is called here the elastic linear system.*

Solution of the system given by eq. (4.2) is a minimizer of problem (4.1). Moreover, it is also useful to identify the matrix on the left-hand side of eq. (4.2) and the vector on the right-hand side of eq. (4.2). Therefore consider the following definition

**Definition 4.3** (Elastic Matrix and Elastic Observation)**.** *Let*

$$\boldsymbol{A} := \boldsymbol{X}^H\boldsymbol{X} + \sum_{j\in\{1,2,\ldots,K\}} \mu_j \boldsymbol{B}_j^T \boldsymbol{B}_j \tag{4.3}$$

*be the elastic matrix and*

$$\boldsymbol{b} := \boldsymbol{X}^H\boldsymbol{y} + \sum_{j\in\{1,2,\ldots,K\}} \mu_j \boldsymbol{B}_j^T \boldsymbol{d}_{(j)} - \boldsymbol{v} \tag{4.4}$$

*be the elastic observation. Note that, the elastic matrix is symmetric in the sense described in Section 3.4.*

In Chapter 7 it is more clear why the names were chosen in this way. In addition, we also assume that all operators, vectors and parameters are known from the context.

## 4.2 Efficient Way of Solving the Elastic Linear System

The solution of the system given by eq. (4.2) can be obtained if we know the inverse of the elastic matrix (Definition 4.3). Although in general it might not be possible to solve this system since the system depends on the operators $\boldsymbol{X}$ and $\boldsymbol{B}_j$ for $j \in \{1, 2, \ldots, K\}$, in this thesis we have to only deal with the operators, the subsampling Fourier operator $\boldsymbol{X}$, an orthonormal wavelet operator $\boldsymbol{B}_a$, the finite difference operator $\boldsymbol{B}_r$ and the imaginary part penalizing operator $\boldsymbol{B}_i$, defined in Section 3.2. Moreover, we consider the set of chosen indices $\mathcal{J}$ containing those indices which correspond to cutting off columns in the Fourier space. Therefore, in the context of our interest

$$\boldsymbol{I}_{\cdot,\mathcal{J}}\boldsymbol{I}_{\mathcal{J},\cdot} = \boldsymbol{I}_{j\in\mathcal{J}}$$

where $\boldsymbol{I}_{\mathcal{J},\cdot}$ is the subsampling operator, $\boldsymbol{I}_{\cdot,\mathcal{J}}$ is the upsampling operator[3], $\boldsymbol{I}_{j\in\mathcal{J}} := diag(\delta_{j\in\mathcal{J}})$ and $\delta_{j\in\mathcal{J}}$ is the Kronecker delta defined as

$$\delta_{j\in\mathcal{J}} := \begin{cases} 1 & \text{if } j \in \mathcal{J} \\ 0 & \text{otherwise} \end{cases}$$

[3] Definition 3.1 and Definition 3.2.

Moreover, the elastic matrix has the following form

$$\boldsymbol{A} := \boldsymbol{X}^H \boldsymbol{X} + \sum_{j \in \{a,r,i\}} \mu_j \boldsymbol{B}_j^T \boldsymbol{B}_j \tag{4.5}$$

and the elastic observation is

$$\boldsymbol{b} := \boldsymbol{X}^H \boldsymbol{y} + \sum_{j \in \{a,r,i\}} \mu_j \boldsymbol{B}_j^T \boldsymbol{d}_{(j)} - \boldsymbol{v} \tag{4.6}$$

Under this definition of the elastic matrix and the elastic observation we have to show how the following system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \tag{4.7}$$

can be solved.

We have

$$\begin{aligned}
&\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \\
&(\boldsymbol{X}^H \boldsymbol{X} + \mu_a \boldsymbol{B}_a^T \boldsymbol{B}_a + \mu_r \boldsymbol{B}_r^T \boldsymbol{B}_r + \mu_i \boldsymbol{B}_i^T \boldsymbol{B}_i)\boldsymbol{u} = \boldsymbol{b} \\
&(\boldsymbol{X}^H \boldsymbol{X} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{B}_r^T \boldsymbol{B}_r + \mu_i \boldsymbol{B}_i^T \boldsymbol{B}_i)\boldsymbol{u} = \boldsymbol{b} \\
&\boldsymbol{F}(\boldsymbol{X}^H \boldsymbol{X} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{B}_r^T \boldsymbol{B}_r + \mu_i \boldsymbol{B}_i^T \boldsymbol{B}_i)\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b} \\
&(\boldsymbol{I}_{\cdot,\mathcal{J}} \boldsymbol{I}_{\mathcal{J},\cdot} \boldsymbol{F} + \mu_a \boldsymbol{F} + \mu_r \boldsymbol{F}\boldsymbol{B}_r^T (\boldsymbol{F}\boldsymbol{B}_r^T)^T \boldsymbol{F} + \mu_i \boldsymbol{F}(\boldsymbol{B}_i^T \boldsymbol{B}_i)\boldsymbol{F}^H \boldsymbol{F})\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b} \\
&(\boldsymbol{I}_{j \in \mathcal{J}} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{D} + \mu_i \boldsymbol{D}_i)\boldsymbol{F}\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b} \\
&\bar{\boldsymbol{D}}\boldsymbol{F}\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b}
\end{aligned} \tag{4.8}$$

where

$$\begin{aligned}
\boldsymbol{I}_{j \in \mathcal{J}} &:= diag(\delta_{j \in \mathcal{J}}) \\
\boldsymbol{D} &:= \boldsymbol{F}\boldsymbol{B}_r^T (\boldsymbol{F}\boldsymbol{B}_r^T)^T \\
\boldsymbol{D}_i &:= \boldsymbol{F}\boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{F}^H \\
\bar{\boldsymbol{D}} &:= (\boldsymbol{I}_{j \in \mathcal{J}} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{D} + \mu_i \boldsymbol{D}_i)
\end{aligned}$$

Therefore the solution of the linear system given by eq. (4.7) is

$$\begin{aligned}
&\boldsymbol{u} = \boldsymbol{F}^H \hat{\boldsymbol{D}} \boldsymbol{F} \boldsymbol{b} \\
\text{where} \quad &\hat{\boldsymbol{D}} := (\boldsymbol{I}_{j \in \mathcal{J}} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{D} + \mu_i \boldsymbol{D}_i)^{-1}
\end{aligned} \tag{4.9}$$

and we can conclude that

$$\boldsymbol{A}^{-1} = \boldsymbol{F}^H \hat{\boldsymbol{D}} \boldsymbol{F} \tag{4.10}$$

It only remains to show how the matrix $\hat{\boldsymbol{D}}$ can be computed. Let start from the following matrix $\boldsymbol{D}_f := \boldsymbol{I}_{j \in \mathcal{J}} + \mu_a \boldsymbol{I} + \mu_r \boldsymbol{D}$. Since

$$\boldsymbol{B}_r^T = \left[ \boldsymbol{P}_h^T - \boldsymbol{I},\ \boldsymbol{P}_v^T - \boldsymbol{I} \right]$$

where (see Subsection 3.2.4 for more details)

$$\begin{aligned}
(\boldsymbol{P}_h^T \boldsymbol{u})^{\mathcal{M}}_{(y,x)} &= \boldsymbol{u}^{\mathcal{M}}_{(y,[x-1]_{n_x})} \\
(\boldsymbol{P}_v^T \boldsymbol{u})^{\mathcal{M}}_{(y,x)} &= \boldsymbol{u}^{\mathcal{M}}_{([y-1]_{n_y},x)}
\end{aligned}$$

by the virtue of shift theorem we have

$$(\boldsymbol{F}\boldsymbol{P}_h^T\boldsymbol{u})^{\mathcal{M}}_{(\omega,\gamma)} = e^{-i2\pi\frac{\gamma}{n_x}}(\boldsymbol{F}\boldsymbol{u})^{\mathcal{M}}_{(\omega,\gamma)}$$

and

$$(\boldsymbol{F}\boldsymbol{P}_v^T\boldsymbol{u})^{\mathcal{M}}_{(\omega,\gamma)} = e^{-i2\pi\frac{\omega}{n_y}}(\boldsymbol{F}\boldsymbol{u})^{\mathcal{M}}_{(\omega,\gamma)}$$

Therefore

$$\begin{aligned}\boldsymbol{D} &= \boldsymbol{F}\left[\boldsymbol{P}_h^T - \boldsymbol{I},\ \boldsymbol{P}_v^T - \boldsymbol{I}\right](\boldsymbol{F}\left[\boldsymbol{P}_h^T - \boldsymbol{I},\ \boldsymbol{P}_v^T - \boldsymbol{I}\right])^T \\ &= \left[diag(e^{-i2\pi\frac{\gamma}{n_x}} - 1)\boldsymbol{F} \quad diag(e^{-i2\pi\frac{\omega}{n_y}} - 1)\boldsymbol{F}\right]\begin{bmatrix}\boldsymbol{F}^H diag(e^{-i2\pi\frac{\gamma}{n_x}} - 1)^H \\ \boldsymbol{F}^H diag(e^{-i2\pi\frac{\omega}{n_y}} - 1)^H\end{bmatrix} \\ &= diag(|e^{-i2\pi\frac{\gamma}{n_x}} - 1|^2) + diag(|e^{-i2\pi\frac{\omega}{n_y}} - 1|^2)\end{aligned} \tag{4.11}$$

Summarizing $\boldsymbol{D}_f = \boldsymbol{I}_{j\in\mathcal{J}} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{D}$ is a diagonal matrix. Moreover, because $\mu_a > 0$, $\mu_r > 0$ and $\mu_i > 0$ all diagonal entries of the matrix are strictly positive and so the matrix itself has full rank[4]. Note that the operator $\boldsymbol{D}_f$ can also be seen as the following function in the Fourier domain

$$(\boldsymbol{D}_f\boldsymbol{u})^{\mathcal{M}}_{(\omega,\gamma)} = (\delta_{\gamma\in\mathcal{J}} + \mu_a + \mu_r(|e^{-i2\pi\frac{\omega}{n_y}} - 1|^2 + |e^{-i2\pi\frac{\gamma}{n_x}} - 1|^2))\boldsymbol{u}^{\mathcal{M}}_{(\omega,\gamma)} \tag{4.12}$$

Let $\boldsymbol{v} \in \mathbb{C}^n$ be a complex-valued vector and $n := n_x \cdot n_y$. Now the matrix $\boldsymbol{B}_i^T\boldsymbol{B}_i = \boldsymbol{I}\otimes\boldsymbol{\delta}_2\boldsymbol{\delta}_2^T$ gains the interpretation of the operator which leaves the imaginary part intact, that is

$$\boldsymbol{B}_i^T\boldsymbol{B}_i\{\boldsymbol{v} := \boldsymbol{a} + i\boldsymbol{b}\} = i\boldsymbol{b} \tag{4.13}$$

Under this interpretation we have

$$\frac{\boldsymbol{F}^H\boldsymbol{v} - \overline{\boldsymbol{F}^H\boldsymbol{v}}}{2} = \boldsymbol{B}_i^T\boldsymbol{B}_i(\boldsymbol{F}^H\boldsymbol{v}) \tag{4.14}$$

where $\overline{\boldsymbol{v}}$ is the complex conjugate. Let define the reverse operation in the following way

$$\boldsymbol{v}^{\mathcal{M}}_R := [v_{[\boldsymbol{n}-\boldsymbol{k}]_{\boldsymbol{n}}}]^{\boldsymbol{n}-(1,1)}_{\boldsymbol{k}=(0,0)} \tag{4.15}$$

where $\boldsymbol{n} := (n_y, n_x)$, $\boldsymbol{k} := (y, x)$ and $v_{[(a,b)]_{(n_a,n_b)}}$ is the cycling operator[5]. We may also use notation $\boldsymbol{v}_R$ to refer to (4.15). Notice that the reverse operation is a self-invertible and linear operation, that is both $(\boldsymbol{v}_R)_R = \boldsymbol{v}$ and $(\alpha\boldsymbol{v} + \beta\boldsymbol{u})_R = \alpha(\boldsymbol{v})_R + \beta(\boldsymbol{u})_R$ hold (Lemma D.3 and Lemma D.4).

Since $\overline{\boldsymbol{F}^H\boldsymbol{v}} = \boldsymbol{F}^H\overline{\boldsymbol{v}_R}$ holds (Lemma D.2) we can establish equivalence between eq. (4.14) and

$$\boldsymbol{F}^H\frac{\boldsymbol{v} - \overline{\boldsymbol{v}_R}}{2} = \boldsymbol{B}_i^T\boldsymbol{B}_i(\boldsymbol{F}^H\boldsymbol{v}) \tag{4.16}$$

Let $\boldsymbol{P}_a$ be a linear operator defined as

$$\boldsymbol{P}_a\boldsymbol{v} := \frac{\boldsymbol{v} - \overline{\boldsymbol{v}_R}}{2}$$

[4] The embedding $\boldsymbol{D}_f := \boldsymbol{D}_f \otimes \boldsymbol{I}_2$ (see eq. (3.5)) preserves the invertibility of this matrix.

[5] $v_{[(a,b)]_{(n_a,n_b)}} := v_{(a \mod n_a,\ b \mod n_b)}$.

By plugging $\boldsymbol{P}_a$ into eq. (4.16) yields the following equation

$$\boldsymbol{F}^H(\boldsymbol{P}_a\boldsymbol{v}) = \boldsymbol{B}_i^T\boldsymbol{B}_i(\boldsymbol{F}^H\boldsymbol{v}) \tag{4.17}$$

Therefore

$$\boldsymbol{P}_a\boldsymbol{v} = \boldsymbol{F}\boldsymbol{B}_i^T\boldsymbol{B}_i\boldsymbol{F}^H\boldsymbol{v}$$
$$\boldsymbol{P}_a\boldsymbol{v} = \boldsymbol{D}_i\boldsymbol{v}$$

and we can conclude that

$$\boldsymbol{P}_a = \boldsymbol{D}_i \tag{4.18}$$

Interestingly, the left hand side of eq. (4.18) doesn't require any Fourier transform to be performed, and so $\boldsymbol{P}_a\boldsymbol{v}$ can be efficiently computed. The following lemma shows that $\boldsymbol{P}_a$ is a projection

**Lemma 4.1.** *Let $\boldsymbol{P}_a$ be a linear operator defined as*

$$\boldsymbol{P}_a\boldsymbol{v} := \frac{\boldsymbol{v} - \overline{\boldsymbol{v}_R}}{2}$$

*Then the following holds*

$$\forall_{\boldsymbol{v}\in\mathbb{C}^n}\ \boldsymbol{P}_a(\boldsymbol{P}_a\boldsymbol{v}) = \boldsymbol{P}_a\boldsymbol{v}$$

*That is $\boldsymbol{P}_a$ is a projection.*

*Proof.* Pick up $\boldsymbol{v}\in\mathbb{C}^n$. Then we have $\boldsymbol{P}_a\boldsymbol{v} = \frac{\boldsymbol{v}-\overline{\boldsymbol{v}_R}}{2}$. Moreover,

$$\boldsymbol{P}_a(\boldsymbol{P}_a\boldsymbol{v}) = \frac{\boldsymbol{P}_a\boldsymbol{v} - \overline{(\boldsymbol{P}_a\boldsymbol{v})_R}}{2} = \frac{\boldsymbol{v} - \overline{\boldsymbol{v}_R} - \overline{\boldsymbol{v}_R} + (\boldsymbol{v}_R)_R}{4} = \frac{\boldsymbol{v} - \overline{\boldsymbol{v}_R}}{2} = \boldsymbol{P}_a\boldsymbol{v}$$

where we used the property that the operator $(\cdot)_R$ is self-invertible and linear. □

Define $\boldsymbol{P}_h := \frac{\boldsymbol{v}+\overline{\boldsymbol{v}_R}}{2}$. Likewise $\boldsymbol{P}_a$ the operator $\boldsymbol{P}_h$ is also a projection. Because we have $\overline{\boldsymbol{P}_a\boldsymbol{v}} = -(\boldsymbol{P}_a\boldsymbol{v})_R$ and $\overline{\boldsymbol{P}_h\boldsymbol{v}} = (\boldsymbol{P}_h\boldsymbol{v})_R$ we call $\boldsymbol{P}_a$ the projection onto antihermitians and $\boldsymbol{P}_h$ the projection onto hermitians.

Since $(\boldsymbol{P}_a + \boldsymbol{P}_h)\boldsymbol{v} = \boldsymbol{v}$ and $\boldsymbol{P}_a(\boldsymbol{P}_h\boldsymbol{v}) = \boldsymbol{P}_h(\boldsymbol{P}_a\boldsymbol{v}) = \boldsymbol{0}$ every complex-valued vector $\boldsymbol{v}\in\mathbb{C}^n$ can be decomposed in $\boldsymbol{v} = \boldsymbol{v}_a + \boldsymbol{v}_h$ where $\boldsymbol{v}_a := \boldsymbol{P}_a\boldsymbol{v}$ and $\boldsymbol{v}_h := \boldsymbol{P}_h\boldsymbol{v}$. We call the equation $\boldsymbol{v} = \boldsymbol{v}_a + \boldsymbol{v}_h$ the decomposition in antihermitians and hermitians. This is the orthogonal decomposition in the sense that the antihermitian $\boldsymbol{v}_a$ is orthogonal to the hermitian $\boldsymbol{v}_h$, or in other words the equation $\boldsymbol{v}_h^T\boldsymbol{v}_a = 0$ holds.

Consider the following equation

$$(\boldsymbol{D}_f + \mu_i\boldsymbol{P}_a)\boldsymbol{F}\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b} \tag{4.19}$$

where $\boldsymbol{D}_f = \boldsymbol{I}_{j\in\mathcal{J}} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{D}$ is a diagonal matrix with strictly positive entries[6]. Since $\boldsymbol{P}_a = \boldsymbol{D}_i$ holds both equations. (4.19) and (4.8) are equivalent. Let $\boldsymbol{f} := \boldsymbol{F}\boldsymbol{u}$ and $\boldsymbol{f} = \boldsymbol{f}_a + \boldsymbol{f}_h$ be the decomposition in antihermitians and hermitians. Since $\boldsymbol{P}_a$ is the projection onto antihermitians we have $\boldsymbol{P}_a\boldsymbol{f} = \boldsymbol{P}_a(\boldsymbol{f}_a + \boldsymbol{f}_h) = \boldsymbol{f}_a$. Similarly, we have $\boldsymbol{P}_h\boldsymbol{f} = \boldsymbol{P}_h(\boldsymbol{f}_a + \boldsymbol{f}_h) =$

[6] See discussion under eq. (4.11).

$\boldsymbol{f}_h$. Let $\boldsymbol{r} := \boldsymbol{F}\boldsymbol{b}$ and $\boldsymbol{r} = \boldsymbol{r}_a + \boldsymbol{r}_h$ be the decomposition in antihermitians and hermitians. Now, we can establish the following equation

$$\boldsymbol{D}_f \boldsymbol{f} + \mu_i \boldsymbol{f}_a = \boldsymbol{r}_a + \boldsymbol{r}_h \tag{4.20}$$

Eq. (4.20) is equivalent to eq. (4.19). In addition, let

$$\boldsymbol{D}_f = \boldsymbol{D}_a + \boldsymbol{D}_h$$

where $\boldsymbol{D}_f = \operatorname{diag} \boldsymbol{d}$, $\boldsymbol{D}_a := \operatorname{diag} \boldsymbol{d}_a$, $\boldsymbol{D}_h := \operatorname{diag} \boldsymbol{d}_h$, $\boldsymbol{d}_a := \boldsymbol{P}_a \boldsymbol{d}$ and $\boldsymbol{d}_h := \boldsymbol{P}_h \boldsymbol{d}$. Therefore

$$\begin{aligned} \boldsymbol{D}_f \boldsymbol{f} + \mu_i \boldsymbol{f}_a &= (\boldsymbol{D}_a + \boldsymbol{D}_h)(\boldsymbol{f}_a + \boldsymbol{f}_h) + \mu_i \boldsymbol{f}_a \\ = \boldsymbol{D}_a \boldsymbol{f}_a + \boldsymbol{D}_a \boldsymbol{f}_h + \boldsymbol{D}_h \boldsymbol{f}_a + \boldsymbol{D}_h \boldsymbol{f}_h + \mu_i \boldsymbol{f}_a &= \boldsymbol{r}_a + \boldsymbol{r}_h \end{aligned}$$

where $\boldsymbol{r}_a$ is the antihermitian and $\boldsymbol{r}_h$ is the hermitian. Moreover, both $\boldsymbol{D}_h \boldsymbol{f}_h$, $\boldsymbol{D}_a \boldsymbol{f}_a$ are hermitians and both $\boldsymbol{D}_h \boldsymbol{f}_a$, $\boldsymbol{D}_a \boldsymbol{f}_h$ are antihermitians. Therefore we can establish the following two relations

$$\boldsymbol{D}_h \boldsymbol{f}_h + \boldsymbol{D}_a \boldsymbol{f}_a = \boldsymbol{r}_h \tag{4.21}$$

$$\boldsymbol{D}_a \boldsymbol{f}_h + \boldsymbol{D}_h \boldsymbol{f}_a + \mu_i \boldsymbol{f}_a = \boldsymbol{r}_a \tag{4.22}$$

From eq. (4.22) we have

$$\boldsymbol{f}_a = (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} (\boldsymbol{r}_a - \boldsymbol{D}_a \boldsymbol{f}_h) \tag{4.23}$$

Computing inverse of the matrix $\boldsymbol{D}_h + \mu_i \boldsymbol{I}$ is allowed because it is a diagonal matrix with strictly positive entries. Plugging eq. (4.23) into eq. (4.21) yields

$$\boldsymbol{f}_h = (\boldsymbol{D}_h - \boldsymbol{D}_a (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_a)^{-1} (\boldsymbol{r}_h - \boldsymbol{D}_a (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{r}_a) \tag{4.24}$$

Since the matrix $\boldsymbol{D}_f$ is a diagonal matrix with strictly positive entries, we have $\boldsymbol{D}_h \succcurlyeq \boldsymbol{D}_a$ where $\succcurlyeq$ denotes the component-wise weak inequality $\geq$. Moreover, since

$$a > \frac{a^2}{a + \mu} \text{ for any } \mu > 0$$

the following also holds

$$\boldsymbol{D}_h \succ \boldsymbol{D}_h (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_h$$

where $\succ$ denotes the component-wise strong inequality $>$. Therefore

$$\boldsymbol{D}_h - \boldsymbol{D}_a (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_a \succcurlyeq \boldsymbol{D}_h - \boldsymbol{D}_h (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_h \succ \boldsymbol{0}$$

and so computing inverse of the matrix $\boldsymbol{D}_h - \boldsymbol{D}_a (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_a$ is allowed. Moreover, computing $\boldsymbol{f}_a$ and $\boldsymbol{f}_h$ is not only possible but it is also tractable because both matrices $\boldsymbol{D}_h + \mu_i \boldsymbol{I}$ and $\boldsymbol{D}_h - \boldsymbol{D}_a (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1} \boldsymbol{D}_a$ are diagonal and so their inverse can be efficiently computed. Then $\boldsymbol{u}$ can be recovered from the following equation

$$\boldsymbol{f}_a + \boldsymbol{f}_h = \boldsymbol{F}\boldsymbol{u} \tag{4.25}$$

In summary, the solution of the linear system (4.7) is

$$\boldsymbol{u} = \boldsymbol{F}^H (\boldsymbol{f}_a + \boldsymbol{f}_h) \tag{4.26}$$

where

$$\boldsymbol{r_a} := \boldsymbol{P}_a \boldsymbol{F} \boldsymbol{b}$$

$$\boldsymbol{r_h} := \boldsymbol{P}_h \boldsymbol{F} \boldsymbol{b}$$

$$\boldsymbol{f}_h = (\boldsymbol{D}_h - \boldsymbol{D}_a(\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1}\boldsymbol{D}_a)^{-1}(\boldsymbol{r}_h - \boldsymbol{D}_a(\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1}\boldsymbol{r}_a)$$

and

$$\boldsymbol{f}_a = (\boldsymbol{D}_h + \mu_i \boldsymbol{I})^{-1}(\boldsymbol{r}_a - \boldsymbol{D}_a \boldsymbol{f}_h)$$

This shows how the linear system given by eq. (4.2) can be solved in the context of our interest. Moreover, the space and time complexity of applying operators $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$, $\boldsymbol{P}_a$, $\boldsymbol{P}_h$ and $\boldsymbol{X}$ to any vector $\boldsymbol{v}$ is at most linearithmic with respect to the number of entries of the vector $\boldsymbol{v}$, and the complexity of computing the inverse of any diagonal matrix is in worst case linear. Therefore the solution given by eq. (4.26) can also be computed in the linearithmic time and space complexity.

## 4.3 Efficient Way of Solving the Elastic Linear System - Different Set of the Penalizing Operators

One can consider the objective function (Definition 3.11) with different penalizing operators. If obtained elastic linear system satisfies some conditions, briefly discussed in this section, this linear system can still be efficiently solved. Similarly to eq. (4.8), we have

$$(\boldsymbol{I}_{j\in\mathcal{J}} + \sum_j \mu_j \boldsymbol{D}_j)\boldsymbol{F}\boldsymbol{u} = \boldsymbol{F}\boldsymbol{b} \tag{4.27}$$

where $\boldsymbol{D}_j := \boldsymbol{F}(\boldsymbol{B}_j^T \boldsymbol{B}_j)\boldsymbol{F}^H$. Note that, if there exist $k$ such that $\boldsymbol{D}_k$ is a diagonal matrix with strictly positive entries, and for every $j \neq k$ the matrix $\boldsymbol{D}_j$ is a diagonal matrix with non-negative entries, then $(\boldsymbol{I}_{j\in\mathcal{J}} + \sum_j \mu_j \boldsymbol{D}_j)$ is also a diagonal matrix with strictly positive entries, and so the system (4.27) can be efficiently solved. The solution is

$$\boldsymbol{u} = \boldsymbol{F}^H(\boldsymbol{I}_{j\in\mathcal{J}} + \sum_j \mu_j \boldsymbol{D}_j)^{-1}\boldsymbol{F}\boldsymbol{b}$$

To conclude, although in this thesis we mainly focus on three penalizing operators $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$, one can consider different set of the penalizing operators, and use algorithms presented in Chapter 6 and Chapter 7 assuming that for every $\boldsymbol{v}$ the efficient way of performing the following operation

$$(\boldsymbol{I}_{j\in\mathcal{J}} + \sum_j \mu_j \boldsymbol{D}_j)^{-1}\boldsymbol{v}$$

is provided.

Chapter 5

# Proximity Operator

## Contents



Most algorithms described in this thesis split the original problem to subproblems that can be efficiently solved by using a projection or proximity operators. This chapter provides definitions and show some instances of these operators that play crucial roles in this thesis.

In Section 5.1, two important operators, a projection operator and a proximity operator, are defined. In Section 5.2, a few special cases of the proximity operators are shown. Finally, in Section 5.3, the soft-thresholding operator is introduced. This operator applied to a vector comes into play as the solution of one of the subproblems.

## 5.1 From Projection to Proximity Operator

This section starts from the projection operator defined as

**Definition 5.1** (Projection onto $\mathcal{K}$)**.** *Assume that $\mathcal{K} \neq \emptyset$ is a closed and convex subset of $\mathbb{R}^n$. For every $\boldsymbol{y} \in \mathbb{R}^n$ let*

$$P_{\mathcal{K}}(\boldsymbol{y}) := \arg\min_{\boldsymbol{u} \in \mathcal{K}} \frac{1}{2}||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 \tag{5.1}$$

*The operator $P_{\mathcal{K}}(\cdot)$ is called the projection operator.*

Figure 5.1 shows geometrical interpretation of this concept. The projection operator is the solution of the following problem

$$\min_{\boldsymbol{u} \in \mathcal{K}} \frac{1}{2}||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2$$

where $\mathcal{K} \neq \emptyset$ is some closed and convex subset of $\mathbb{R}^n$. The same problem can be also formulated as an unconstrained problem

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} \iota_{\mathcal{K}}(\boldsymbol{u}) + \frac{1}{2}||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2$$

where $\iota_{\mathcal{K}}(\cdot)$ is the indicator function of the set $\mathcal{K}$ defined as

$$\iota_{\mathcal{K}}(\boldsymbol{u}) := \begin{cases} 0 & \text{if } \boldsymbol{u} \in \mathcal{K} \\ \infty & \text{otherwise} \end{cases}$$

Although the indicator function $\iota_{\mathcal{K}}$ is used nothing prevents us from using other functions. This leads to the generalization of the projection operator known as the proximity operator

**Definition 5.2** (Proximity Operator)**.** *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a continuous and convex function. For every $\boldsymbol{y} \in \mathbb{R}^n$ let consider the following minimization problem*

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\text{minimize}} \frac{1}{2}||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + g(\boldsymbol{u})$$

*which admits a unique solution denoted by the $prox_g(\boldsymbol{y})$. The operator $prox_g(\cdot)$ is called the proximity operator [Combettes and Pesquet 2010].*

## 5.2 Examples of the Proximity Operator

In this thesis two special cases of the proximity operator are encountered. If we take $g(\cdot) := ||\cdot||_{l_1}$ in the proximity operator, we get a so called denoising problem defined as

**Definition 5.3** (Denoising Problem)**.** *Let consider, for fixed $\boldsymbol{y} \in \mathbb{R}^n$, the following minimization problem*

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \lambda||\boldsymbol{u}||_{l_1} \tag{5.2}$$

*This problem is called the denoising problem.*

We can also consider another version of the denoising problem with TV-seminorm instead of $L_1$-norm, that is

**Definition 5.4** (TV-based Denoising Problem)**.** *Let consider, for fixed $\boldsymbol{y} \in \mathbb{R}^n$, the following minimization problem*

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \lambda||\boldsymbol{u}||_{TV} \tag{5.3}$$

*This problem is called the TV-based denoising problem.*

The problem related to the TV-based denoising problem is

**Definition 5.5** (TV-based Deblurring Problem)**.** *For every $\boldsymbol{y} \in \mathbb{R}^n$ let*

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \lambda||\boldsymbol{u}||_{TV} \tag{5.4}$$

*where $\boldsymbol{X} : \mathbb{R}^n \to \mathbb{R}^m$ is some linear operator. This problem is called the TV-based deblurring problem.*

## 5.3 Soft-Thresholding Operator

In this section we introduce soft-thresholding operator which can be used to obtain the solution of the denoising problem. Soft-thresholding operator shrinks all coefficients of the vector above the threshold in absolute value and kills completely other coefficients (see also Figure 5.2). Formally this operator can be defined as follows

**Definition 5.6** (Soft-Thresholding Operator)**.** *For every non-zero $z \in \mathbb{R}$ or $z \in \mathbb{C}$ let*

$$S_\tau(z) := \frac{z}{|z|}(|z| - \tau)_+$$

*where*

$$(x)_+ := \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

*and $\tau$ is the threshold. In addition we may consider the following extension to vectors. Let $\boldsymbol{z} \in \mathbb{R}^n$ or $\boldsymbol{z} \in \mathbb{C}^n$. Then*

$$S_\tau(\boldsymbol{z}) := [S_\tau(z_i)]_{i \in \mathcal{I}}$$

*where $\mathcal{I}$ is the set of indices of the vector $\boldsymbol{z}$. The operator $S_\tau(\cdot)$ is called the soft-thresholding operator. This operator can also be readily extended to work with the real-valued surrogate of the complex field, it is enough to define*

$$|\boldsymbol{z}| := ||\boldsymbol{z}||_{l_2} \text{ for every } \boldsymbol{z} \in \mathcal{C}$$

*where $\mathcal{C} := \mathbb{R}^2$.*

The following theorem establishes relationship between the proximity operator with $||\cdot||_{l_1}$ and the soft-thresholding operator. It also shows that the soft-thresholding operator is the solution of the denoising problem (Definition 5.3).

**Theorem 5.1.** *The soft-thresholding operator is the solution of the denoising problem. More precisely, the following holds*

$$S_\tau(\boldsymbol{z}) = prox_{\tau||\cdot||_{l_1}}(\boldsymbol{z})$$

*Proof.* Proof can be found in Combettes and Wajs [2005]. □

By using Proposition 3.1 in Combettes and Wajs [2005] we can also deduce the following statement

**Corollary 5.1.** *Consider the following problem*

$$\arg\min_{\boldsymbol{u} \in \mathbb{R}^n} \frac{1}{2}||\boldsymbol{G}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \tau||\boldsymbol{u}||_{l_1} \tag{5.5}$$

*where $\boldsymbol{G}$ is some matrix. Let $\boldsymbol{u}^\star \in \mathbb{R}^n$. Then the following two statements are equivalent*

- *$\boldsymbol{u}^\star$ is a solution of problem 5.5*
- *$\boldsymbol{u}^\star = prox_{\gamma\tau||\cdot||_{l_1}}(\boldsymbol{u}^\star - \gamma\boldsymbol{G}^T(\boldsymbol{G}\boldsymbol{u}^\star - \boldsymbol{y})) = S_{\gamma\tau}(\boldsymbol{u}^\star - \gamma\boldsymbol{G}^T(\boldsymbol{G}\boldsymbol{u}^\star - \boldsymbol{y}))$*

*where $\frac{1}{\gamma} \geq ||\boldsymbol{G}^T\boldsymbol{G}||_{l_2}$.*

Also worth mentioning is that the fixed-point equation provided in Corollary 5.1 can be used to establish the fixed-point iteration in the *Constant-step Forward-backward* algorithm [Combettes and Pesquet 2010].

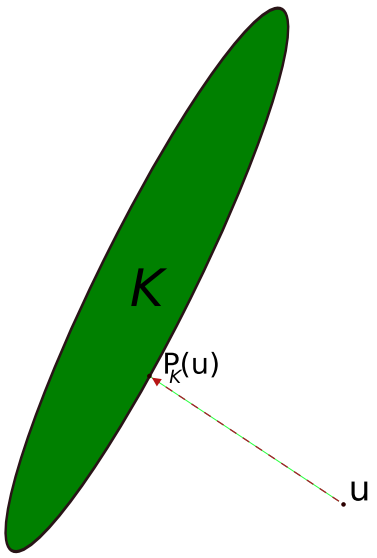



Figure 5.1: Geometrical interpretation of the projection operator. Intuitively, the solution of the projection of a point $\boldsymbol{u}$ onto set $\mathcal{K}$ is the point $P_{\mathcal{K}}(\boldsymbol{u})$ in set $\mathcal{K}$ which is the closest to $\boldsymbol{u}$ among all points belonging to $\mathcal{K}$.

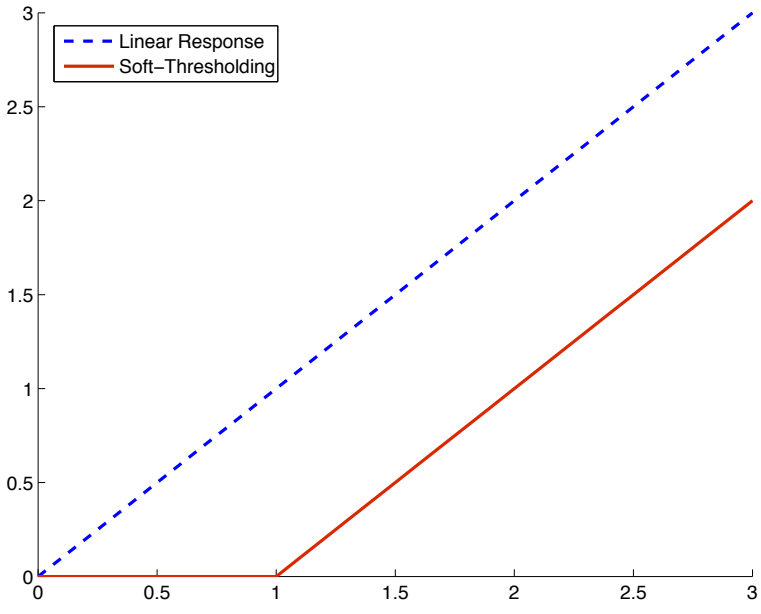



Figure 5.2: Figure shows linear response and soft-thresholding response $S_1(\cdot)$ to one dimensional variables in interval between 0 and 3. Threshold was set to be $\tau := 1$.

Chapter 6

# First Order Methods

## Contents



In this chapter, two first order methods that are used in the comparison are derived. Both deal with the problem of finding the minimizer of the MRI energy function (Definition 3.11). Furthermore, both algorithms have been derived from different frameworks. First method comes from the *FISTA* framework. The second method comes from the *Augmented Lagrangian* framework. In addition, the third algorithm used in the comparison called *Non-linear Conjugate Gradient* is briefly shown.

Section 6.1 defines the differentiable surrogate of the $l_1$-norm which is used to modify the energy function. Next, the *Non-linear Conjugate Gradient* method is briefly presented. Finally, the gradient of the modified energy function is derived. Section 6.2 describes *Steepest Descent* which is a simple algorithm suitable for solving unconstrained problems with convex and differentiable objective function. Next, *Gradient Projection* and *Fast Gradient Projection* are presented. Both can be seen as an extension of *Steepest Descent* suitable for solving optimization problems constrained to closed and convex feasible sets. Subsequently, *FISTA*

and *TV-FISTA* are introduced. The former is a general method which exploits the proximity operator in order to solve an optimization problem, latter was derived for problems with the total-variation regularization. Finally, $l_1$-*FISTA* is shown, a first order method that can be used in order to solve problems with the regularization term of kind $\sum_j ||\boldsymbol{B}_j \boldsymbol{u}||_{l_1}$. Section 6.3 starts with the *Variable Splitting* and the *Quadratic Penalty* approaches. The former method utilizes splitting in order to decouple the objective function by introducing new variables and coupling them together. The main idea of the *Quadratic Penalty* approach is transforming the equality constrained optimization problem into unconstrained one by introducing a special function which penalizes for violating the equality constraints. The *Augmented Lagrangian* approach augments the *Quadratic Penalty* approach with the Lagrangian in order to obtain a method with better behavior. Finally, method based on the *Variable Splitting* approach and the *Augmented Lagrangian* approach is derived. We also show application of this method to the problem of finding the minimizer of the MRI energy function.

## 6.1 Non-linear Conjugate Gradient

### 6.1.1 Differentiable Surrogate of MRI Energy Function

Since the MRI energy function (3.6) is not differentiable, in order to apply the *Non-linear Conjugate Gradient* method this energy function has to be modified

**Definition 6.1** (Differentiable Surrogate of the MRI Energy Function)**.** *Let $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$, $\boldsymbol{X}$ be, respectively, the wavelet transform, the finite difference operator, the imaginary part penalizing operator and the subsampling Fourier operator, and assume that all of them are tailored to work with the real-valued surrogate of the complex set $\mathcal{C} := \mathbb{R}^2$. Let $||\cdot||_\varepsilon$ be the differentiable surrogate of the $l_1$-norm defined as*

$$||\boldsymbol{s}||_\varepsilon := \sum_j \sqrt{||\boldsymbol{s}_j||_{l_2}^2 + \varepsilon} \tag{6.1}$$

*for some $\varepsilon > 0$ and every $\boldsymbol{s} \in \mathcal{C}^n$. For every $\boldsymbol{u} \in \mathcal{C}^n$ let*

$$\Psi_\varepsilon^{MRI}(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_\varepsilon + \kappa_r||\boldsymbol{B}_r\boldsymbol{u}||_\varepsilon + \kappa_i||\boldsymbol{B}_i\boldsymbol{u}||_\varepsilon \tag{6.2}$$

*where for each $j \in \{a, r, i\}$ $\kappa_j := \tau_j\sigma$, $\tau_j$ is a model parameter, $\sigma^2$ is the noise variance. The function $\Psi^{MRI}(\cdot)$ is called the differentiable surrogate of the MRI energy function. Operators $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$ are also called penalizing operators.*

Note that the differentiable surrogate of the $l_1$-norm is an upper bound of the latter, that is

$$||\boldsymbol{s}||_{l_1} = \sum_j ||\boldsymbol{s}_j||_{l_2} < \sum_j \sqrt{||\boldsymbol{s}_j||_{l_2}^2 + \varepsilon} = ||\boldsymbol{s}||_\varepsilon \text{ for every } \varepsilon > 0$$

Therefore, if $\varepsilon$ gets smaller this upper bound becomes more tight. However, in practice, we may run in numerical troubles if we decide to pick too small $\varepsilon$.

### 6.1.2 Algorithm

**Algorithm 1** Non-linear Conjugate Gradient Algorithm for Problem (3.7)

**Require:** Initial value $\boldsymbol{u}^0$

$t_{\max} := 1$
**for** $k = 0, 1, \ldots$ **do**
  $(f, \boldsymbol{g}) := \left(\Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}^k), \nabla\Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}^k)\right)$
  $gSqNorm := ||\boldsymbol{g}||_{l_2}^2$
  **if** $k == 0$ **then**
    $\boldsymbol{d} := -\boldsymbol{g}$
  **else**
    $\nu := \frac{gSqNorm}{gSqNormOld + \varepsilon_{machine}}$ where $\varepsilon_{machine}$ is the machine precision
    $\boldsymbol{d} := -\boldsymbol{g} + \nu\boldsymbol{d}$
    **if** $\boldsymbol{g}^T\boldsymbol{d} \geq 0$ **then**
      Restart the search direction: $\boldsymbol{d} := -\boldsymbol{g}$
    **end if**
  **end if**
  $gSqNormOld := gSqNorm$
  {Line-search on the line $\left\{\boldsymbol{u}^k + \alpha\boldsymbol{d}\right\}_\alpha$:}
  $t_r := \boldsymbol{g}^T\boldsymbol{d}$
  $t_i := (\Re\boldsymbol{g})^T\Im\boldsymbol{d} - (\Im\boldsymbol{g})^T\Re\boldsymbol{d}$ where $\Re\boldsymbol{s} := \left(\boldsymbol{I} \otimes \boldsymbol{\delta}_1^T\right)\boldsymbol{s}$ and $\Im\boldsymbol{s} := \left(\boldsymbol{I} \otimes \boldsymbol{\delta}_2^T\right)\boldsymbol{s}$
  $r := 0.01\sqrt{t_r^2 + t_i^2}$
  $\alpha := t_{\max}$
  $f_n = \Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}^k + \alpha\boldsymbol{d})$
  $l := 1$
  **while** $f_n > f - \alpha r$ and $l < l_{\max}$ **do**
    $\alpha := \beta \cdot \alpha$
    $f_n = \Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}^k + \alpha\boldsymbol{d})$
    $l := l + 1$
  **end while**
  {We found good $\alpha$}
  {Update solution:}
  $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha\boldsymbol{d}$
  {Adapt initial step size:}
  **if** $l > l_{above}$ **then**
    $t_{\max} := \beta \cdot t_{\max}$
  **else if** $l == l_{below}$ **then**
    $t_{\max} := \frac{t_{\max}}{\beta}$
  **end if**
  **if** stopping criterion is satisfied **then**
    break the loop
  **end if**
**end for**

Algorithm 1 is a standard *Non-linear Conjugate Gradient* algorithm tuned to our setting [Nocedal and Wright 1999]. Similar algorithm was also used by Lustig et al. [2007]. In our setup $l_{\max} := 20$, $\beta := 0.6$, $l_{above} := 3$, $l_{below} := 1$ and $\varepsilon := 4 \cdot 10^{-12}$. We also chose the maximal number of iterations as a stopping criterion.

In order to obtain good efficiency of Algorithm 1 we have to minimize the number of invocations of the fast Fourier transform which means that for any $\boldsymbol{s}$ the number of occurrences of $\boldsymbol{X}\boldsymbol{s}$ and $\boldsymbol{X}^H\boldsymbol{s}$ should be minimal. Let

$$\psi_{reg}(\boldsymbol{u}) := \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_\varepsilon + \kappa_r||\boldsymbol{B}_r\boldsymbol{u}||_\varepsilon + \kappa_i||\boldsymbol{B}_i\boldsymbol{u}||_\varepsilon \tag{6.3}$$

Note that

$$\begin{aligned}
&\Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}+\alpha\boldsymbol{d}) \\
&= \frac{1}{2}||\boldsymbol{X}\left(\boldsymbol{u}+\alpha\boldsymbol{d}\right)-\boldsymbol{y}||_{l_2}^2 + \psi_{reg}(\boldsymbol{u}+\alpha\boldsymbol{d}) \\
&= \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \frac{\alpha^2}{2}||\boldsymbol{X}\boldsymbol{d}||_{l_2}^2 + \alpha\left(\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}\right)^T\boldsymbol{d} + \psi_{reg}(\boldsymbol{u}+\alpha\boldsymbol{d}) \\
&= \frac{1}{2}||\boldsymbol{r}||_{l_2}^2 + \frac{\alpha^2}{2}||\boldsymbol{X}\boldsymbol{d}||_{l_2}^2 + \alpha\boldsymbol{r}^T\boldsymbol{d} + \psi_{reg}(\boldsymbol{u}+\alpha\boldsymbol{d})
\end{aligned}$$

where $\boldsymbol{r} := \boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}$. In addition, the update step for $\boldsymbol{u}^k$ is

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha\boldsymbol{d}$$

and can be transformed into

$$\boldsymbol{r}^{k+1} = \boldsymbol{r}^k + \alpha\boldsymbol{X}\boldsymbol{d}$$

where $\boldsymbol{r}^k := \boldsymbol{X}\boldsymbol{u}^k-\boldsymbol{y}$. Similarly, we have

$$\begin{aligned}
&\nabla\Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}) \\
&= \boldsymbol{X}^H\left(\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}\right) + \nabla\psi_{reg}(\boldsymbol{u}) \\
&= \boldsymbol{X}^H\boldsymbol{r} + \nabla\psi_{reg}(\boldsymbol{u})
\end{aligned}$$

This means that if we compute $\boldsymbol{r}^0 := \boldsymbol{X}\boldsymbol{u}^0-\boldsymbol{y}$ before iterations start and maintain $\boldsymbol{r}^k$ along the iterations, this algorithm requires only two invocations of the fast Fourier transform per iteration: to compute $\boldsymbol{X}\boldsymbol{d}$ and to compute $\boldsymbol{X}^H\boldsymbol{r}$.

Nocedal and Wright [1999] contains good introduction to both *Conjugate Gradient* algorithm and *Non-linear Conjugate Gradient*.

### 6.1.3 Gradient

Recall the energy function used in the *Non-linear Conjugate Gradient* algorithm

$$\Psi_\varepsilon^{\mathrm{MRI}}(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \psi_{reg}(\boldsymbol{u}) \tag{6.4}$$

where

$$\psi_{reg}(\boldsymbol{u}) := \kappa_a||\boldsymbol{B}_a\boldsymbol{u}||_\varepsilon + \kappa_r||\boldsymbol{B}_r\boldsymbol{u}||_\varepsilon + \kappa_i||\boldsymbol{B}_i\boldsymbol{u}||_\varepsilon$$

We have to figure out gradient of the energy function given by eq. (6.4) in order to run the *Non-linear Conjugate Gradient* algorithm. We have

$$\nabla \Psi_{\varepsilon}^{\mathrm{MRI}}(\boldsymbol{u}) = \boldsymbol{X}^H (\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}) + \nabla \psi_{reg}(\boldsymbol{u})$$

Let focus on $\nabla \psi_{reg}(\boldsymbol{u})$. For every $\boldsymbol{s}_j \in \mathcal{C}$, define

$$|\boldsymbol{s}_j|_{\varepsilon} := \sqrt{||\boldsymbol{s}_j||_{l_2}^2 + \varepsilon} \tag{6.5}$$

Notice that given (6.5), for every $\boldsymbol{s} \in \mathcal{C}^n$, the differentiable surrogate of the $l_1$-norm can be expressed as

$$||\boldsymbol{s}||_{\varepsilon} = \sum_j |\boldsymbol{s}_j|_{\varepsilon}$$

Let $\boldsymbol{e}_j$ be the $j$-th standard vector in $\mathcal{C}^n$ (the standard vectors were described in Appendix E). Since $\nabla_{\boldsymbol{s}_j} |\boldsymbol{s}_j|_{\varepsilon} = \frac{\boldsymbol{s}_j}{|\boldsymbol{s}_j|_{\varepsilon}}$ we have

$$\nabla_{\boldsymbol{s}} ||\boldsymbol{s}||_{\varepsilon} = \sum_j \frac{1}{|\boldsymbol{s}_j|_{\varepsilon}} \boldsymbol{e}_j \boldsymbol{s}_j \tag{6.6}$$

Put $\boldsymbol{s} := \boldsymbol{B}\boldsymbol{u}$. Because of (6.6) we have (see also Appendix F)

$$\nabla_{\boldsymbol{u}} ||\boldsymbol{s}||_{\varepsilon} = \nabla_{\boldsymbol{u}} ||\boldsymbol{B}\boldsymbol{u}||_{\varepsilon} = \sum_j \frac{1}{|\boldsymbol{s}_j|_{\varepsilon}} \boldsymbol{B}^T \boldsymbol{e}_j \boldsymbol{s}_j = \boldsymbol{B}^T \left(\boldsymbol{W}^{-1} \otimes \boldsymbol{I}_2\right) \boldsymbol{B}\boldsymbol{u} \tag{6.7}$$

where $\boldsymbol{W}$ is a diagonal matrix with the following diagonal elements

$$\boldsymbol{W}_{j,j} := \sqrt{||(\boldsymbol{B}\boldsymbol{u})_j||_{l_2}^2 + \varepsilon}$$

Equipped with eq. (6.7) we can easily compute the whole gradient of $\Psi_{\varepsilon}^{\mathrm{MRI}}(\cdot)$.

## 6.2 FISTA

### 6.2.1 Gradient Projection

**Steepest Descent.** Consider the following unconstrained minimization problem

$$\boldsymbol{u}^{\star} = \arg\min_{\boldsymbol{u}} f(\boldsymbol{u}) \tag{6.8}$$

where it is assumed that $f$ is a convex and continuously differentiable function. A natural algorithm that solves problem (6.8) could be the one which constructs a sequence $\{\boldsymbol{u}_k\}_k$ of decreasing objective values, that is

$$\forall_k \forall_{j<k}\ f(\boldsymbol{u}_k) < f(\boldsymbol{u}_j)$$

In this context it means that directional derivative

$$f'(\boldsymbol{u}; \boldsymbol{d}) := \lim_{h \to 0} \frac{f(\boldsymbol{u} + h\boldsymbol{d}) - f(\boldsymbol{u})}{h} \quad \text{where } ||d||_{l_2} \leq 1$$

is negative. This suggests the following relation between two consecutive points

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha_k \boldsymbol{d}^k$$

where $\alpha_k$ is called the step-size and $\boldsymbol{d}^k$ is a descent direction. We have to also take a special care of the choice of step-size. Otherwise the algorithm may not converge. Now we can establish the following optimization problem to find the direction

$$\boldsymbol{d}^k = \underset{||\boldsymbol{d}||_{l_2} \leq 1}{\arg\min} f'(\boldsymbol{u}^k; \boldsymbol{d})$$

and since $f'(\boldsymbol{u}; \boldsymbol{d}) = \nabla f(\boldsymbol{u})^T \boldsymbol{d}$ also holds it follows that $\boldsymbol{d}^k := -\frac{\nabla f(\boldsymbol{u}^k)}{||\nabla f(\boldsymbol{u}^k)||_{l_2}}$. The whole algorithm can be summarized as follows

**Algorithm 2** Steepest Descent

**Require:** $\boldsymbol{u}^0$

1: **for** $k = 0, 1, \ldots$ **do**
2: $\quad \boldsymbol{d}^k = -\nabla f(\boldsymbol{u}^k)$
3: $\quad$ Perform line-search, that is $\alpha_k \in \arg\min_{\alpha > 0} \; f(\boldsymbol{u}^k + \alpha \boldsymbol{d}^k)$
4: $\quad \boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha_k \boldsymbol{d}^k$
5: $\quad$ **if** stopping criterion is satisfied **then**
6: $\quad\quad$ break the loop
7: $\quad$ **end if**
8: **end for**

There are a few methods to find the step-size $\alpha_k$ in Algorithm 2, for example it can be found analytically or by using a backtracking line-search[1].

We can use the maximal allowed number of iterations as the stopping criterion. Another possibility is to monitor the two consecutive solutions and stop whenever the difference is small enough. More precisely, the algorithm stops when the condition $||\boldsymbol{u}^{k+1} - \boldsymbol{u}^k||_{l_2} < \varepsilon_{stop}$ holds for some chosen in advance $\varepsilon_{stop}$.

Intuition behind the algorithm is straightforward, we want to find a descent direction $\boldsymbol{d}$ and then minimize the objective function along the line given by $\boldsymbol{d}$.

**Gradient Projection.** Consider the following optimization problem

$$\boldsymbol{u}^\star = \underset{\boldsymbol{u} \in \mathcal{K}}{\arg\min} f(\boldsymbol{u}) \tag{6.9}$$

where $\mathcal{K}$ is a closed and convex set, $f$ is a convex differentiable function with Lipschitz-continuous gradient and with Lipschitz constant $L$, that is

$$\forall_{\boldsymbol{u},\boldsymbol{v}} \; ||\nabla f(\boldsymbol{u}) - \nabla f(\boldsymbol{v})||_{l_2} \leq L||\boldsymbol{u} - \boldsymbol{v}||_{l_2} \tag{6.10}$$

Lipschitz-continuity imposed on the gradient implies [Beck and Teboulle 2009a]

$$\forall_{\boldsymbol{u},\boldsymbol{v}} \; f(\boldsymbol{u}) \leq f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^T (\boldsymbol{u} - \boldsymbol{v}) + \frac{L}{2}||\boldsymbol{u} - \boldsymbol{v}||_{l_2}^2 \tag{6.11}$$

Keeping fixed $\boldsymbol{v} := \boldsymbol{u}^k$ and minimizing over $\boldsymbol{u}$ yields the *Gradient Projection* algorithm with the following fixed point iteration

$$\boldsymbol{u}^{k+1} = P_{\mathcal{K}}(\boldsymbol{u}^k - \alpha \nabla f(\boldsymbol{u}^k)) \tag{6.12}$$

[1] Two backtracking line-search algorithms are presented in Appendix G.

where $\alpha$ is such that $L \leq \frac{1}{\alpha}$ and $P_{\mathcal{K}}$ is a projection onto the set $\mathcal{K}$ (Definition 5.1).

Intuitively, the update step given by eq. (6.12) can be explained as follows. When the descent direction $\boldsymbol{d}^k = -\nabla f(\boldsymbol{u}^k)$ and the step-size $\alpha$ are chosen the algorithm makes a move to the new point $\boldsymbol{u}^+ := \boldsymbol{u}^k + \alpha \boldsymbol{d}^k$. If the point $\boldsymbol{u}^+$ is feasible then $\boldsymbol{u}^{k+1} := \boldsymbol{u}^+$, otherwise the projection $P_{\mathcal{K}}$ warrants the new point $\boldsymbol{u}^{k+1}$ to be feasible by taking $\boldsymbol{u}^{k+1} := P_{\mathcal{K}}(\boldsymbol{u}^+)$.

Now we can establish the following algorithm

**Algorithm 3** Gradient Projection

**Require:** $\boldsymbol{u}^0$, $\alpha > 0$

1: **for** $k = 0, 1, \ldots$ **do**
2: $\quad \boldsymbol{u}^{k+1} = P_{\mathcal{K}}(\boldsymbol{u}^k - \alpha \nabla f(\boldsymbol{u}^k))$
3: $\quad$ **if** stopping criterion is satisfied **then**
4: $\quad\quad$ break the loop
5: $\quad$ **end if**
6: **end for**

where $P_{\mathcal{P}}(\boldsymbol{u})$ is the projection of $\boldsymbol{u}$ onto the set $\mathcal{K}$. The order of the rate of convergence of this algorithm is $O(1/k)$ [Beck and Teboulle 2009b].

**Fast Gradient Projection.** *Fast Gradient Projection* (FGP) warrants better theoretical rate of convergence than *Gradient Projection*. The order of rate of convergence of FGP is $O(1/k^2)$ [Beck and Teboulle 2009b]. Fixed point iteration of FGP can be expressed as follows

$$\begin{aligned} \boldsymbol{u}^k &= P_{\mathcal{K}}(\boldsymbol{v}^k - \alpha \nabla f(\boldsymbol{v}^k)) \\ t_{k+1} &= \frac{1 + \sqrt{1 + 4\, t_k^2}}{2} \\ \boldsymbol{v}^{k+1} &= \boldsymbol{u}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}) \end{aligned}$$

So speeding up was gained by taking the update step for $\boldsymbol{v}^{k+1}$ to be a linear combination of two feasible points $\boldsymbol{u}^k$ and $\boldsymbol{u}^{k-1}$.

This idea can be summarized in the form of the following algorithm

**Algorithm 4** Fast Gradient Projection

**Require:** $\boldsymbol{u}^0$, $\boldsymbol{v}^1$ , $t_1$ (by default $t_1 := 1$), $\alpha > 0$

1: **for** $k = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{u}^k = P_{\mathcal{K}}(\boldsymbol{v}^k - \alpha \nabla f(\boldsymbol{v}^k))$
3: $\quad$ **if** stopping criterion is satisfied **then**
4: $\quad\quad$ break the loop
5: $\quad$ **else**
6: $\quad\quad t_{k+1} = \frac{1+\sqrt{1+4\, t_k^2}}{2}$
7: $\quad\quad \boldsymbol{v}^{k+1} = \boldsymbol{u}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{u}^k - \boldsymbol{u}^{k-1})$
8: $\quad$ **end if**
9: **end for**

where $P_{\mathcal{K}}(\boldsymbol{u})$ is the projection of $\boldsymbol{u}$ onto the set $\mathcal{K}$.

### 6.2.2 Introduction to FISTA

**FISTA.** *Fast Iterative Shrinkage/Thresholding Algorithm*, or shorter *FISTA*, was introduced to deal with the following optimization problem

$$\underset{\boldsymbol{u}}{\text{minimize}} \{F(\boldsymbol{u}) := f(\boldsymbol{u}) + g(\boldsymbol{u})\}$$

where it is assumed that both functions $f$ and $g$ are continuous and convex. In addition it is assumed that $f$ is continuously differentiable and has Lipschitz-continuous gradient with Lipschitz-constant $L$, that is $\forall_{\boldsymbol{u},\boldsymbol{v}}\ ||\nabla f(\boldsymbol{u}) - \nabla f(\boldsymbol{v})||_{l_2} \leq L||\boldsymbol{u} - \boldsymbol{v}||_{l_2}$.

In Beck and Teboulle [2009a] fixed point iteration of *FISTA* was proposed to be

$$\begin{aligned} \boldsymbol{u}^k &= prox_{\alpha g}(\boldsymbol{v}^k - \alpha \nabla f(\boldsymbol{v}^k)) \\ t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ \boldsymbol{v}^{k+1} &= \boldsymbol{u}^k + (\frac{t_k - 1}{t_{k+1}})(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}) \end{aligned}$$

where $\alpha \leq \frac{1}{L}$ and *prox* is the proximity operator (Definition 5.2).

The algorithm is similar to *Fast Gradient Projection* (Algorithm 4) and can be summarized as follows

**Algorithm 5** FISTA

**Require:** $\boldsymbol{u}^0$, $\boldsymbol{v}^1$ , $t_1$ (by default $t_1 := 1$)

1: **for** $k = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{u}^k = prox_{\alpha g}(\boldsymbol{v}^k - \alpha \nabla f(\boldsymbol{v}^k))$
3: $\quad$ **if** stopping criterion is satisfied **then**
4: $\quad\quad$ break the loop
5: $\quad$ **else**
6: $\quad\quad t_{k+1} = \frac{1+\sqrt{1+4\,t_k^2}}{2}$
7: $\quad\quad \boldsymbol{v}^{k+1} = \boldsymbol{u}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{u}^k - \boldsymbol{u}^{k-1})$
8: $\quad$ **end if**
9: **end for**

It was proven that *FISTA* exhibits the global convergence rate of order $O(1/k^2)$ [Beck and Teboulle 2009a]. Although *FISTA* is a general method the major difficulty is to find the proximity operator $prox_{\alpha g}(\cdot)$ in the closed-form. Note also that the difference between Algorithm 4 and Algorithm 5 is that the former uses projection whereas the latter uses the proximity operator.

### 6.2.3 Dual Norm

Consider the $l_p$-norm given by eq. (1.4). For every $\boldsymbol{u} \in \mathbb{R}^n$ the dual of the $l_p$-norm is

$$||\boldsymbol{u}||_p^* := \underset{\boldsymbol{v} \in \mathbb{R}^n}{\text{maximize}} \left\{ \boldsymbol{u}^T \boldsymbol{v} \mid ||\boldsymbol{v}||_p \leq 1 \right\} \tag{6.13}$$

In this thesis we focus only on the $l_2$-norm. It can be shown that the $l_2$-norm is self-dual [Boyd and Vandenberghe 2004], that is

$$||\cdot||_{l_2}^* = ||\cdot||_{l_2} \tag{6.14}$$

We will use eq. (6.14) in Subsection 6.2.4 to derive the dual of the TV-seminorm.

### 6.2.4 Dual of the TV-seminorm

In this subsection, we show the dual form of the TV-seminorm. Later, we will use it in Subsection 6.2.5 to derive *TV-FISTA*. Let start from the definition of P-Set

**Definition 6.2** (P-Set)**.** *Let $\mathcal{C} := \mathbb{R}^2$, the following set*

$$\mathcal{P}(n) = \left\{\boldsymbol{p} \in \mathcal{C}^n \mid ||\boldsymbol{p}_j||_{l_2} \leq 1\right\}$$

*is the P-Set. If dimensionality of the problem can be inferred from the context we may use $\mathcal{P} := \mathcal{P}(n)$ to simplify notation.*

Equipped with the definition of P-Set we can show the dual of the TV-seminorm

**Theorem 6.1** (Dual of the TV-seminorm)**.** *Let $T(\boldsymbol{u}, \boldsymbol{p}) = \sum_i \boldsymbol{p}_i^T \boldsymbol{s}_i$, $\boldsymbol{s} = \boldsymbol{B}_r \boldsymbol{u}$ and $\mathcal{P}$ be P-Set. Then we have*

$$||\boldsymbol{u}||_{TV} = \underset{\boldsymbol{p} \in \mathcal{P}}{\text{maximize}}\, T(\boldsymbol{u}, \boldsymbol{p})$$

*where $||\boldsymbol{u}||_{TV} := ||\boldsymbol{B}_r \boldsymbol{u}||_{l_1}$ (Subsection 3.2.5).*

*Proof.* By duality of the $l_2$-norm we have

$$||\boldsymbol{u}||_{TV} = \sum_i ||\boldsymbol{s}_i||_{l_2} = \sum_i \underset{||\boldsymbol{p}_i||_{l_2} \leq 1}{\text{maximize}}\, \boldsymbol{p}_i^T \boldsymbol{s}_i = \underset{\boldsymbol{p} \in \mathcal{P}}{\text{maximize}} \sum_i \boldsymbol{p}_i^T \boldsymbol{s}_i = \underset{\boldsymbol{p} \in \mathcal{P}}{\text{maximize}}\, T(\boldsymbol{u}, \boldsymbol{p})$$

□

Therefore the TV-seminorm can be reformulated as an optimization problem.

### 6.2.5 TV-FISTA

Consider the following TV-based deblurring problem

$$\underset{\boldsymbol{u}}{\text{minimize}} \left\{E(\boldsymbol{u}) := ||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + 2\lambda ||\boldsymbol{u}||_{TV}\right\} \tag{6.15}$$

where $||\boldsymbol{u}||_{TV} := ||\boldsymbol{B}_r \boldsymbol{u}||_{l_1}$. Finding the solution of problem (6.15) is not straightforward; there are two main problems:

- The energy function $E(\cdot)$ is not differentiable, so we cannot use classic optimization methods such as *Steepest Descent* which exploit information about the gradient of $E(\cdot)$.
- Matrix $\boldsymbol{B}_r$ is not invertible, so we cannot use directly the soft-thresholding operator to obtain the solution.

*TV-FISTA* is an algorithm which tries to solve problem (6.15) by exploiting dual of the TV-seminorm.

The following theorem allows us, under some assumptions, to exchange the minimization with the maximization

**Theorem 6.2** (Minimax Theorem)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be non-empty closed convex sets in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, and let function $f$ be continuous finite concave-convex function on $\mathcal{A} \times \mathcal{B}$. If either $\mathcal{A}$ or $\mathcal{B}$ is bounded then*

$$\inf_{\boldsymbol{b}\in\mathcal{B}} \sup_{\boldsymbol{a}\in\mathcal{A}} f(\boldsymbol{a},\boldsymbol{b}) = \sup_{\boldsymbol{a}\in\mathcal{A}} \inf_{\boldsymbol{b}\in\mathcal{B}} f(\boldsymbol{a},\boldsymbol{b})$$

*Proof.* Proof can be found in Rockafellar [1997] (Corollary 37.3.2). □

**TV-FISTA.** *TV-FISTA* exploits dual approach to solve problem (6.15) [Beck and Teboulle 2009b; Chambolle 2004]. Since our derivation of the $l_1$-*FISTA* is based on the same idea, it is instrumental to derive *TV-FISTA* for the case $\mathcal{K} = \mathcal{C}^n$. Theorem 6.1 tells that

$$||\boldsymbol{u}||_{TV} = \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, T(\boldsymbol{u},\boldsymbol{p})$$

where $T(\boldsymbol{u},\boldsymbol{p}) = \sum_i \boldsymbol{p}_i^T \boldsymbol{B}_r \boldsymbol{u}_i$ and $\boldsymbol{p}_j, \boldsymbol{u}_j \in \mathcal{C}$.

Now

$$\begin{aligned}
&\underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\lambda||\boldsymbol{u}||_{TV} \\
&= \underset{\boldsymbol{u}}{\text{minimize}}\, \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\lambda T(\boldsymbol{u},\boldsymbol{p}) \\
&= \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, \underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\lambda T(\boldsymbol{u},\boldsymbol{p}) \\
&= \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, \underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\lambda \sum_i \boldsymbol{p}_i^T \boldsymbol{B}_r \boldsymbol{u}_i \\
&= \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, \underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\lambda\, \boldsymbol{p}^T \boldsymbol{B}_r \boldsymbol{u} \\
&= \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, -||\boldsymbol{y}-\lambda \boldsymbol{B}_r^T \boldsymbol{p}||_{l_2}^2 + C \\
&= -\underset{\boldsymbol{p}\in\mathcal{P}}{\text{minimize}} \left\{ h(\boldsymbol{p}) := ||\boldsymbol{y}-\lambda \boldsymbol{B}_r^T \boldsymbol{p}||_{l_2}^2 \right\} + C
\end{aligned}$$

where we used Theorem 6.2 to exchange the minimization with the maximization, and where $\sum_i \boldsymbol{a}_i^T \boldsymbol{b}_i = \boldsymbol{a}^T \boldsymbol{b}$ for $\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathcal{C}$, and $C$ is some term independent of neither $\boldsymbol{u}$ nor $\boldsymbol{p}$. Notice that, due to the duality, we can transform the non-differentiable objective function into the differentiable one. However, nothing is given freely. It was done at the expense of additional constraints.

Now we can use the *Gradient Projection* approach to solve $\arg\min_{\boldsymbol{p}\in\mathcal{P}} h(\boldsymbol{p})$. Fixed point iteration becomes

$$\boldsymbol{p}^{k+1} = P_{\mathcal{P}}(\boldsymbol{p}^k - \alpha \nabla h(\boldsymbol{p}^k))$$

It is easy to see that[2]

$$\nabla h(\boldsymbol{p}^k) = -2\lambda \boldsymbol{B}_r (\boldsymbol{y} - \lambda \boldsymbol{B}_r^T \boldsymbol{p})$$

[2] See also Appendix F.

We have to also figure out the value of $\alpha$. Notice that $\alpha$ is such that $L \leq \frac{1}{\alpha}$. Since $L \leq 16\lambda^2$ (see Lemma 4.2 in Beck and Teboulle [2009b]) we can set $\alpha := \frac{1}{16\lambda^2}$, and thus the fixed point iteration becomes now

$$\boldsymbol{p}^{k+1} = P_{\mathcal{P}}(\boldsymbol{p}^k + \frac{1}{8\lambda}\boldsymbol{B}_r(\boldsymbol{y} - \lambda \boldsymbol{B}_r^T \boldsymbol{p}^k))$$

and solution of the primal problem can be obtained from

$$\boldsymbol{u}^\star = \boldsymbol{y} - \lambda \boldsymbol{B}_r^T \boldsymbol{p}^\star$$

All in all the algorithm can be described as

**Algorithm 6** TV-FISTA

**Require:** $\boldsymbol{p}^0$, $\boldsymbol{q}^1$ , $t_1$ (by default $t_1 := 1$), $\alpha > 0$

**for** $k = 1, 2, \ldots$ **do**
  $\boldsymbol{p}^k = P_{\mathcal{P}}(\boldsymbol{q}^k + \frac{1}{8\lambda}\boldsymbol{B}_r(\boldsymbol{y} - \lambda \boldsymbol{B}_r^T \boldsymbol{q}^k))$
  **if** stopping criterion is satisfied **then**
    $\boldsymbol{u}^\star = \boldsymbol{y} - \lambda \boldsymbol{B}_r^T \boldsymbol{p}^k$
    break the loop
  **else**
    $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
    $\boldsymbol{q}^{k+1} = \boldsymbol{p}^k + (\frac{t_k - 1}{t_{k+1}})(\boldsymbol{p}^k - \boldsymbol{p}^{k-1})$
  **end if**
**end for**

where $P_{\mathcal{P}}(\boldsymbol{p})$ is the projection of $\boldsymbol{p}$ onto P-Set.

In Beck and Teboulle [2009b] and Beck and Teboulle [2009a] it was proven that the following holds

$$F(\boldsymbol{u}^k) - F(\boldsymbol{u}^\star) \leq \frac{2\hat{L}||\boldsymbol{u}^0 - \boldsymbol{u}^\star||_{l_2}^2}{(k+1)^2}$$

where $\{\boldsymbol{u}^k\}$ is a sequence generated by the *TV-FISTA* or *FISTA*, $F(\cdot)$ is the objective function and $\hat{L}$ is an upper bound of the Lipschitz constant. Therefore, it is beneficial for the performance of both methods to find as small upper bound of the Lipschitz constant as possible and as close initial value $\boldsymbol{u}^0$ to $\boldsymbol{u}^\star$ as possible.

### 6.2.6 FISTA in the Deblurring Problem

Beck and Teboulle [2009b] also showed how *FISTA* can be used to solve the TV-based deblurring problem defined as

**Definition 6.3** (TV-based Deblurring Problem)**.** *Put*

$$f_{\boldsymbol{y}}(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2$$

*and*

$$g(\boldsymbol{u}) := \lambda||\boldsymbol{u}||_{TV}$$

*where* $\boldsymbol{X} : \mathbb{R}^n \to \mathbb{R}^m$ *is some linear operator. For every* $\boldsymbol{y} \in \mathbb{R}^n$ *let*

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\arg\min}\, f_{\boldsymbol{y}}(\boldsymbol{u}) + g(\boldsymbol{u}) \tag{6.16}$$

*This problem is called the TV-based deblurring problem.*

Since

$$\nabla f_{\boldsymbol{y}}(\boldsymbol{u}) = \boldsymbol{X}^H (\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y})$$

and the subproblem

$$prox_{\alpha||\cdot||_{TV}}(\boldsymbol{v}^k - \alpha\nabla f_{\boldsymbol{y}}(\boldsymbol{v}^k))$$

can be solved by *TV-FISTA*, we can use *FISTA* presented in Subsection 6.2.2 (Algorithm 5) to solve problem (6.16). The algorithm can be written as follows

**Algorithm 7** FISTA for Problem (6.16)

**Require:** $\boldsymbol{u}^0$, $\boldsymbol{v}^1$ , $t_1$ (by default $t_1 := 1$), $\alpha > 0$

**for** $k = 1, 2, \ldots$ **do**
  $\boldsymbol{u}^k = prox_{\alpha||\cdot||_{TV}}\left(\boldsymbol{v}^k - \alpha\boldsymbol{X}^H\left(\boldsymbol{X}\boldsymbol{v}^k - \boldsymbol{y}\right)\right)$
  **if** stopping criterion is satisfied **then**
    break the loop
  **else**
    $t_{k+1} = \frac{1+\sqrt{1+4\,t_k^2}}{2}$
    $\boldsymbol{v}^{k+1} = \boldsymbol{u}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{u}^k - \boldsymbol{u}^{k-1})$
  **end if**
**end for**

Algorithm 7 is nested. It contains the outer iterations where $\boldsymbol{u}^k$ is updated and the inner iterations where the subproblem $prox_{\alpha||\cdot||_{TV}}(\boldsymbol{v}^k - \alpha\nabla f_{\boldsymbol{y}}(\boldsymbol{v}^k))$ is solved by *TV-FISTA*.

### 6.2.7 $l_1$-FISTA

Consider the following minimization problem

$$\underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + 2\sum_j \kappa_j ||\boldsymbol{B}_j\boldsymbol{u}||_{l_1}$$

Let $\mathcal{P} := \left\{\boldsymbol{p} = [\boldsymbol{p}_{(j)}]_j \mid \boldsymbol{p}_{(j)} \in \mathcal{P}_j\right\}$ where $\mathcal{P}_j := \{\boldsymbol{p} \mid ||p_i||_{l_2} \leq 1\}$ is a P-set. Note that the set $\mathcal{P}$ is also a P-set. Like before we have $||\boldsymbol{s}_{(j)}||_{l_1} = \text{maximize}_{\boldsymbol{p}\in\mathcal{P}}\, T(\boldsymbol{s}_{(j)}, \boldsymbol{p})$ where $\boldsymbol{s}_{(j)} := \boldsymbol{B}_j\boldsymbol{u}$.

Put $\boldsymbol{v} := \sum_j \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)}$, then we have

$$\begin{aligned}
&\underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\sum_j \kappa_j ||\boldsymbol{B}_j \boldsymbol{u}||_{l_1} \\
&= \underset{\boldsymbol{u}}{\text{minimize}}\, \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\sum_j \kappa_j \boldsymbol{p}_{(j)}^T \boldsymbol{B}_j \boldsymbol{u} \\
&= \underset{\boldsymbol{p}\in\mathcal{P}}{\text{maximize}}\, \underset{\boldsymbol{u}}{\text{minimize}}\, ||\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + 2\, \boldsymbol{v}^T \boldsymbol{u} \\
&= -\underset{\boldsymbol{p}\in\mathcal{P}}{\text{minimize}}\, ||\boldsymbol{y}-\boldsymbol{v}||_{l_2}^2 + C \\
&= -\underset{\boldsymbol{p}\in\mathcal{P}}{\text{minimize}} \left\{ h(\boldsymbol{p}) := ||\boldsymbol{y} - \sum_j \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)}||_{l_2}^2 \right\} + C
\end{aligned}$$

where $C$ is some term which is independent of neither $\boldsymbol{u}$ nor $\boldsymbol{p}$. Note that

$$\nabla_k h(\boldsymbol{p}) = \nabla_k ||\boldsymbol{y} - \sum_j \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)}||_{l_2}^2 = -2\kappa_k \boldsymbol{B}_k (\boldsymbol{y} - \sum_j \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)})$$

So the whole gradient of $h(\boldsymbol{p})$ is

$$\nabla h(\boldsymbol{p}) = -2\boldsymbol{B}(\boldsymbol{y} - \boldsymbol{B}^T \boldsymbol{p})$$

where $\boldsymbol{B} = [\kappa_j \boldsymbol{B}_j]_j$ and $\boldsymbol{p} = \left[\boldsymbol{p}_{(j)}\right]_j$.

This idea is transformed into the following algorithm

**Algorithm 8** $l_1$-FISTA

**Require:** Constant $\alpha$ such that $\frac{1}{\alpha} \geq L$, $\boldsymbol{p}^0$, $\boldsymbol{q}^1$, $t_1$ (by default $t_1 := 1$), $\alpha > 0$

**for** $k = 1, 2, \ldots$ **do**
  $\boldsymbol{p}^k = P_{\mathcal{P}}(\boldsymbol{q}^k + 2\alpha \boldsymbol{B}(\boldsymbol{y} - \boldsymbol{B}^T \boldsymbol{q}^k))$
  **if** stopping criterion is satisfied **then**
    $\boldsymbol{u}^\star = \boldsymbol{y} - \boldsymbol{B}^T \boldsymbol{p}^k$
    break the loop
  **else**
    $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
    $\boldsymbol{q}^{k+1} = \boldsymbol{p}^k + (\frac{t_k - 1}{t_{k+1}})(\boldsymbol{p}^k - \boldsymbol{p}^{k-1})$
  **end if**
**end for**

Note that Algorithm 8 works with the P-set instead of the original space and dimension of this working space can be much higher than dimension of the original space.

### 6.2.8 Projection onto P-set

Let $\mathcal{P}$ be a P-Set (Definition 6.2). Then projection of $\boldsymbol{v} \in \mathcal{C}^n$ onto the P-set can be accomplished by

$$P_{\mathcal{P}}(\boldsymbol{v})_i = \frac{\boldsymbol{v}_i}{\max\{1, ||\boldsymbol{v}_i||_{l_2}\}}$$

Term $\max\{1, \cdot\}$ is necessary since we don't want to touch points that are already in the set. Note that $P_{\mathcal{P}}(\cdot)$ is a component-wise projection onto the $l_2$-ball.

### 6.2.9 $l_1$-FISTA in Minimizing MRI Energy Function

#### 6.2.9.1 Upper Bound of the Lipschitz Constant

In order to be sure that *FISTA* converges we have to find an upper bound of the Lipschitz constant $L$. Subsection 6.2.9.2 shows that in order to obtain this upper bound we actually need the upper bound of the operator norm of the operator $\boldsymbol{B}\boldsymbol{B}^T$ where

$$\boldsymbol{B} := \begin{bmatrix} \kappa_a \boldsymbol{B}_a \\ \kappa_i \boldsymbol{B}_i \\ \kappa_r \boldsymbol{B}_r \end{bmatrix}$$

The following two lemmas are necessary to prove Theorem 6.3 which, in turn, provides the upper bound of $L$

**Lemma 6.1.** *Let $\boldsymbol{B}_r$ be the finite difference operator. Then $||\boldsymbol{B}_r||_{l_2} \leq \sqrt{8}$ holds.*

*Proof.* Proof can be found in Beck and Teboulle [2009b]. □

**Lemma 6.2.** *Let $\boldsymbol{B}_a$ be an orthonormal wavelet transform and $\boldsymbol{B}_i$ the imaginary part penalizing operator. Then we have $||\boldsymbol{B}_a||_{l_2} = 1$ and $||\boldsymbol{B}_i||_{l_2} \leq 1$.*

*Proof.* Pick up $\boldsymbol{v}$ such that $||\boldsymbol{v}||_{l_2} = 1$. Since the wavelet operator $\boldsymbol{B}_a$ is orthonormal, it preserves norm of a vector. So the following holds

$$||\boldsymbol{B}_a \boldsymbol{v}||_{l_2} = ||\boldsymbol{v}||_{l_2} = 1 \tag{6.17}$$

Notice that also the following holds

$$||\boldsymbol{B}_i \boldsymbol{v}||_{l_2}^2 = \sum_j \Im v_j{}^2 \leq \sum_j (\Re v_j{}^2 + \Im v_j{}^2) = ||\boldsymbol{v}||_{l_2}^2 = 1 \tag{6.18}$$

Equation (6.17) and equation (6.18) finish the proof. □

By using Lemma 6.1 and Lemma 6.2 we can prove Theorem 6.3 and derive the appropriate upper bound of the Lipschitz constant

**Theorem 6.3.** *Let*

$$\boldsymbol{B} := \begin{bmatrix} \kappa_a \boldsymbol{B}_a \\ \kappa_i \boldsymbol{B}_i \\ \kappa_r \boldsymbol{B}_r \end{bmatrix}$$

*where $\boldsymbol{B}_a$ is an orthonormal wavelet transform, $\boldsymbol{B}_i$ the imaginary part penalizing operator and $\boldsymbol{B}_r$ the finite difference operator. Then we have*

$$||\boldsymbol{B}\boldsymbol{B}^T||_{l_2} \leq \sqrt{72\, \kappa_{\max}^2 (\kappa_a^2 + \kappa_i^2 + 8\kappa_r^2)}$$

*where $\kappa_{\max} := \max\{\kappa_a, \kappa_i, \kappa_r\}$.*

*Proof.* At the beginning consider a more general case with $K$ operators $\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_K$. Pick up $\boldsymbol{p}$ such that $||\boldsymbol{p}||_{l_2} = 1$ and let $\kappa_{\max} = \max\{\kappa_1, \kappa_2, \ldots, \kappa_K\}$. In addition, let $\boldsymbol{B}_{\max}$ be the operator such that

$$\forall_{k \in \{1,2,\ldots,K\}}\ ||\boldsymbol{B}_k||_{l_2} \leq ||\boldsymbol{B}_{\max}||_{l_2}$$

and

$$\exists_{k\in\{1,2,\dots,K\}}\ \boldsymbol{B}_{\max} = \boldsymbol{B}_k$$

Put $\boldsymbol{v} := \boldsymbol{B}^T\boldsymbol{p} = \sum_{k=1}^{K} \kappa_k \boldsymbol{B}_k^T \boldsymbol{p}_{(k)}$. Then we have

$$\begin{aligned}
||\boldsymbol{v}||_{l_2}^2 &\leq ||\sum_{k=1}^{K} \kappa_{\max} \boldsymbol{B}_k^T \boldsymbol{p}_{(k)}||_{l_2}^2 \\
&= \kappa_{\max}^2 ||\sum_{k=1}^{K} \boldsymbol{B}_k^T \boldsymbol{p}_{(k)}||_{l_2}^2 \\
&\leq \kappa_{\max}^2 \left(\sum_{k=1}^{K} ||\boldsymbol{B}_k^T \boldsymbol{p}_{(k)}||_{l_2}\right)^2 \\
&\leq \kappa_{\max}^2 \left(\sum_{k=1}^{K} ||\boldsymbol{B}_k||_{l_2} ||\boldsymbol{p}_{(k)}||_{l_2}\right)^2 \\
&\leq \kappa_{\max}^2 \left(\sum_{k=1}^{K} ||\boldsymbol{B}_{\max}||_{l_2} ||\boldsymbol{p}_{(k)}||_{l_2}\right)^2 \\
&= \kappa_{\max}^2 ||\boldsymbol{B}_{\max}||_{l_2}^2 \left(\sum_{k=1}^{K} ||\boldsymbol{p}_{(k)}||_{l_2}\right)^2 \\
&\leq \kappa_{\max}^2 ||\boldsymbol{B}_{\max}||_{l_2}^2 K^2
\end{aligned}$$

Therefore, if $\boldsymbol{v} := \sum_{j\in\{a,i,r\}} \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)}$ then

$$||\boldsymbol{v}||_{l_2}^2 \leq 8 \cdot 9\ \kappa_{\max}^2 = 72\ \kappa_{\max}^2$$

and finally

$$\begin{aligned}
||\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{p}||_{l_2}^2 &= \sum_{j\in\{a,i,r\}} ||\kappa_j \boldsymbol{B}_j \boldsymbol{v}||_{l_2}^2 \\
&\leq \sum_{j\in\{a,i,r\}} \kappa_j^2 ||\boldsymbol{B}_j||_{l_2}^2 ||\boldsymbol{v}||_{l_2}^2 \\
&\leq ||\boldsymbol{v}||_{l_2}^2 (\kappa_a^2 + \kappa_i^2 + 8\ \kappa_r^2) \\
&\leq 72\ \kappa_{\max}^2 (\kappa_a^2 + \kappa_i^2 + 8\ \kappa_r^2)
\end{aligned}$$

□

#### 6.2.9.2 Algorithm

We can solve problem (3.7) similarly to the TV-based deblurring problem (see Subsection 6.2.6). Each iteration in the deblurring problem requires the denoising problem to be solved. For the latter we can use $l_1$-*FISTA*. Thus the whole algorithm can be divided into outer iterations where the deblurring problem is being solved and inner iterations where the denoising problem is being solved.

Consider the denoising problem

$$\arg\min_{\boldsymbol{u}} ||\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + 2 \sum_{j\in\{a,r,i\}} \kappa_j ||\boldsymbol{B}_j \boldsymbol{u}||_{l_1}$$

or equivalently (see Subsection 6.2.7)

$$\arg\min_{\boldsymbol{p}\in\mathcal{P}} \left\{ h(\boldsymbol{p}) := ||\boldsymbol{y} - \sum_j \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)}||_{l_2}^2 \right\}$$

where the solution can be obtained from the relation $\boldsymbol{u}^\star = \boldsymbol{y} - \boldsymbol{B}^T\boldsymbol{p}^k$. Put all penalizing operators into one matrix

$$\boldsymbol{B} = \begin{bmatrix} \kappa_a \boldsymbol{B}_a \\ \kappa_i \boldsymbol{B}_i \\ \kappa_r \boldsymbol{B}_r \end{bmatrix}$$

In order to run $l_1$-*FISTA* method we need an upper bound of the Lipschitz constant. Note that

$$||\nabla h(\boldsymbol{p}_1) - \nabla h(\boldsymbol{p}_2)||_{l_2} = 2||\boldsymbol{B}\boldsymbol{B}^T(\boldsymbol{p}_2 - \boldsymbol{p}_1)||_{l_2} \leq 2||\boldsymbol{B}\boldsymbol{B}^T||_{l_2}||\boldsymbol{p}_2 - \boldsymbol{p}_1||_{l_2} \tag{6.19}$$

Therefore we can take $\bar{\alpha}$ which satisfies

$$\frac{1}{\bar{\alpha}} \geq 2||\boldsymbol{B}\boldsymbol{B}^T||_{l_2}$$

Eq. (6.19) warrants that $\frac{1}{\bar{\alpha}} \geq L$ where $L$ is the Lipschitz constant. Following Subsection 6.2.9.1 we can set

$$\bar{\alpha} := 1/(2\sqrt{72\,\kappa_{\max}^2(\kappa_a^2 + \kappa_i^2 + 8\kappa_r^2)})$$

where $\kappa_{\max} := \max\{\kappa_a, \kappa_i, \kappa_r\}$. All in all we have the following update step for $l_1$-*FISTA*

$$\boldsymbol{p}^{(k+1)} = P_{\mathcal{P}}(\boldsymbol{q}^{(k)} + \frac{1}{\sqrt{72\,\kappa_{\max}^2(\kappa_a^2 + \kappa_i^2 + 8\kappa_r^2)}}\boldsymbol{B}(\boldsymbol{y} - \boldsymbol{B}^T\boldsymbol{q}^{(k)})) \tag{6.20}$$

Since the TV-based deblurring problem is the same as introduced in Subsection 6.2.6 we have to only derive an appropriate upper bound $L_D$ of the Lipschitz constant for the measurement matrix $\boldsymbol{X} := \boldsymbol{I}_{\mathcal{J},\cdot}\boldsymbol{F}$. Since we have

$$\begin{aligned} &||\boldsymbol{X}^H(\boldsymbol{X}\boldsymbol{u}_1 - \boldsymbol{y}) - \boldsymbol{X}^H(\boldsymbol{X}\boldsymbol{u}_1 - \boldsymbol{y})||_{l_2} \\ &= ||\boldsymbol{X}^H\boldsymbol{X}(\boldsymbol{u}_1 - \boldsymbol{u}_2)||_{l_2} \\ &\leq ||\boldsymbol{X}^H\boldsymbol{X}||_{l_2}||\boldsymbol{u}_1 - \boldsymbol{u}_2||_{l_2} \\ &\leq ||\boldsymbol{u}_1 - \boldsymbol{u}_2||_{l_2} \end{aligned}$$

we can set $L_D := 1$.

Equipped with all these results we can employ $l_1$-*FISTA* method (Algorithm 8) with $\alpha := \bar{\alpha}$ to minimize the MRI energy function. The algorithm is nested, it performs the inner iterations and the outer iterations, and is stated as follows

**Algorithm 9** FISTA for the MRI energy function

**Require:** $\alpha > 0$ (default $\alpha := \bar{\alpha}$), $L_D > 0$ (default $L_D := 1$), $\boldsymbol{u}^0$, $\boldsymbol{v}^1$, $t_1$ (by default $t_1 := 1$)

1: **for** $k = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{u}^k = D(\boldsymbol{v}^k - \frac{1}{L_D}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{v}^k - \boldsymbol{y}), \frac{\alpha}{L_D})$
3: $\quad$ **if** stopping criterion is satisfied **then**
4: $\quad\quad$ break the loop
5: $\quad$ **end if**
6: $\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
7: $\quad \boldsymbol{v}^{k+1} = \boldsymbol{u}^k + (\frac{t_k - 1}{t_{k+1}})(\boldsymbol{u}^k - \boldsymbol{u}^{k-1})$
8: **end for**

where $D(\boldsymbol{u}, \alpha)$ is $l_1$-*FISTA* (Algorithm 8) with warm start $\boldsymbol{u}$ and step-size parameter $\alpha$. Note that Algorithm 9 needs only two invocations of the Fourier transform per iteration.

## 6.3 Augmented Lagrangian in Minimizing MRI Energy Function

### 6.3.1 Variable Splitting and Quadratic Penalty Approaches

Consider an unconstrained optimization problem such that the objective function can be expressed as a sum of two functions $f_1$, $f_2$ where the second one is a composition of a function and a matrix $\boldsymbol{T}$. More formally, consider the following problem

$$\underset{\boldsymbol{u}\in\mathbb{R}^n}{\text{minimize}}\, f_1(\boldsymbol{u}) + f_2(\boldsymbol{T}\boldsymbol{u}) \tag{6.21}$$

The *Variable Splitting* approach transforms the unconstrained problem (6.21) into the constrained one by adding equality constraints $\boldsymbol{v} = \boldsymbol{T}\boldsymbol{u}$. This leads to the following formula

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{u},\boldsymbol{v}\in\mathbb{R}^n} & f_1(\boldsymbol{u}) + f_2(\boldsymbol{v}) \\ \text{s.t.} & \boldsymbol{v} = \boldsymbol{T}\boldsymbol{u} \end{array} \tag{6.22}$$

The rationale behind splitting is to transform an unconstrained problem into the constrained problem which may happen to be easier to solve by the use of conventional methods, for instance the methods proposed by Beck and Teboulle [2009a] or Bioucas-Dias and Figueiredo [2007]. Splitting can also be helpful in dealing with a linear combination of more than two functions

$$\alpha_1 f_1(\boldsymbol{T}_1\boldsymbol{u}) + \alpha_2 f_2(\boldsymbol{T}_2\boldsymbol{u}) + \alpha_3 f_3(\boldsymbol{T}_3\boldsymbol{u}) + \ldots + \alpha_k f_k(\boldsymbol{T}_k\boldsymbol{u})$$

We can convert problem (6.22) into the unconstrained one by considering the modified version of the objective function. The modified objective function consists of the original objective function $f_1(\boldsymbol{u}) + f_2(\boldsymbol{v})$ and an additional regularization term. The role of this additional term is to penalize the objective function whenever the current point $(\boldsymbol{u}, \boldsymbol{v})$ is not feasible. Such an approach is sometimes called the *Exterior Penalty* approach [Nocedal and Wright 1999]. Here, we consider an instance of the *Exterior Penalty* approach called

*Quadratic Penalty* which utilizes the quadratic penalty function to penalize deviations from being feasible. This function is defined as follows

**Definition 6.4** (Quadratic Penalty Function)**.** *The quadratic penalty function for problem (6.22) is expressed as*

$$Q(\boldsymbol{u}, \boldsymbol{v}; \mu) := f_1(\boldsymbol{u}) + f_2(\boldsymbol{v}) + \frac{\mu}{2}||\boldsymbol{T}\boldsymbol{u} - \boldsymbol{v}||_{l_2}^2$$

*where $\mu > 0$ is the penalty parameter.*

In Bioucas-Dias and Figueiredo [2008] (see also Alfonso et al. [2009]) the problem

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{minimize}}\, Q(\boldsymbol{u}, \boldsymbol{v}; \mu) \tag{6.23}$$

was solved by minimizing $\boldsymbol{Q}$ once over $\boldsymbol{u}$ and once over $\boldsymbol{v}$, and by embedding the penalty parameter $\mu$ into a continuation process where $\mu$ was a slowly increasing parameter. The former can also be seen as a block-coordinate descent algorithm where for fixed $\boldsymbol{u} := \boldsymbol{u}^k$ the problem $\text{minimize}_{\boldsymbol{v}}\, Q(\boldsymbol{u}^k, \boldsymbol{v}; \mu)$ is solved and similarly for fixed $\boldsymbol{v} := \boldsymbol{v}^k$ the problem $\text{minimize}_{\boldsymbol{u}}\, Q(\boldsymbol{u}, \boldsymbol{v}^k; \mu)$ is solved. The continuation process can be intuitively explained as follows: by driving $\mu$ to $\infty$, we penalize the constraints violations more and more severe but at the expense of solving more difficult problems. In Nocedal and Wright [1999] (Section 17.1) these intuitions were formalized and it was proven that under certain conditions, the approximate minimizers $\boldsymbol{u}^k$, $\boldsymbol{v}^k$ of $Q(\boldsymbol{u}, \boldsymbol{v}; \mu_k)$ satisfy

$$\boldsymbol{T}\boldsymbol{u}^k - \boldsymbol{v}^k = \frac{1}{\mu_k}\lambda^\star \tag{6.24}$$

where $\lambda^\star$ is a multiplier that satisfies the KKT[3] conditions (see Theorem 17.2 in Nocedal and Wright [1999] for details). Eq. (6.24) can also be seen as a perturbation in the KKT conditions that vanishes whenever $\mu_k \to \infty$. Note also that for big enough $\mu_k$ we have $\boldsymbol{T}\boldsymbol{u}^k - \boldsymbol{v}^k \approx \boldsymbol{0}$, and so the constraints for problem (6.22) are approximately satisfied.

### 6.3.2 Augmented Lagrangian Method

We start from the augmented Lagrange function which is the main 'building block' of the *Augmented Lagrangian* approach

**Definition 6.5** (Augmented Lagrange Function)**.** *Consider the following optimization problem*

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{u}\in\mathbb{R}^n} & f(\boldsymbol{u}) \\ s.t. & \boldsymbol{A}\boldsymbol{u} - \boldsymbol{y} = \boldsymbol{0} \end{array} \tag{6.25}$$

*The augmented Lagrange function for problem (6.25) is defined as*

$$\mathcal{L}_A(\boldsymbol{u}, \boldsymbol{\lambda}, \mu) = f(\boldsymbol{u}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 \tag{6.26}$$

*where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers and $\mu > 0$ is the penalty parameter.*

[3]Information about the KKT conditions can be found in Nocedal and Wright [1999] or Boyd and Vandenberghe [2004], or in Appendix C.

By the *Augmented Lagrangian* approach we can formulate and solve the following optimization problem

$$\underset{\boldsymbol{\lambda}}{\text{maximize}}\,\underset{\boldsymbol{u}}{\text{minimize}}\;\mathcal{L}_A(\boldsymbol{u},\boldsymbol{\lambda},\mu) \tag{6.27}$$

By comparing gradients of the augmented Lagrange function (6.26) and the Lagrange function (C.2) one can derive the update step for $\boldsymbol{\lambda}$, and the following algorithm for problem (6.27) can be proposed

**Algorithm 10** Augmented Lagrangian Method

**Require:** $\mu_0 > 0, \boldsymbol{\lambda}^0$

**for** $k = 0, 1, \ldots$ **do**
  $\boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}}\; \mathcal{L}_A(\boldsymbol{u}, \boldsymbol{\lambda}^k, \mu_k)$
  **if** stopping criterion is satisfied **then**
    break the loop
  **else**
    $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \mu_k(\boldsymbol{A}\boldsymbol{u}^{k+1} - \boldsymbol{y})$
    Pick up $\mu_{k+1} \in [\mu_k, \infty)$
  **end if**
**end for**

The *Augmented Lagrangian* approach has some interesting properties. As it was pointed in Nocedal and Wright [1999] we have

$$\boldsymbol{A}\boldsymbol{u}^k - \boldsymbol{y} \approx \frac{1}{\mu_k}(\boldsymbol{\lambda}^\star - \boldsymbol{\lambda}^k)$$

so we can conclude that perturbation in the KKT conditions vanishes whenever $\mu_k \to \infty$ or $\boldsymbol{\lambda}^k$ is close enough to the optimal Lagrange multiplier $\boldsymbol{\lambda}^\star$. This observation allows us to fix the penalty parameter $\mu_k$ from the beginning. Precise argument about the convergence of the *Augmented Lagrangian* method (Algorithm 10) for some fixed $\mu > 0$ the reader can find in Nocedal and Wright [1999].

Notice that

$$\begin{aligned} & f(\boldsymbol{u}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 \\ \underset{\hat{\boldsymbol{\lambda}}:=\frac{1}{\mu}\boldsymbol{\lambda}}{\equiv} & f(\boldsymbol{u}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y} + \hat{\boldsymbol{\lambda}}||_{l_2}^2 - \frac{\mu}{2}||\hat{\boldsymbol{\lambda}}||_{l_2}^2 \\ \underset{\boldsymbol{d}:=\boldsymbol{y}-\hat{\boldsymbol{\lambda}}}{\equiv} & f(\boldsymbol{u}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{d}||_{l_2}^2 - \frac{\mu}{2}||\boldsymbol{y} - \boldsymbol{d}||_{l_2}^2 \end{aligned}$$

Since the last term $\frac{\mu}{2}||\boldsymbol{y} - \boldsymbol{d}||_{l_2}^2$ is independent of $\boldsymbol{u}$ assuming fixed $\boldsymbol{d}$, we can derive the following update step for $\boldsymbol{u}$

$$\boldsymbol{u}_{k+1} \in \arg\min_{\boldsymbol{u}}\; f(\boldsymbol{u}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{d}^k||_{l_2}^2$$

The update step for $\boldsymbol{d}$ becomes now

$$\boldsymbol{d}^{k+1} = \boldsymbol{y} + (\boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{u}^{k+1})$$

Now we can derive the alternative version of Algorithm 10

**Algorithm 11** Augmented Lagrangian Method (Alternative Version with Fixed $\mu$)

**Require:** $\mu > 0, \boldsymbol{d}^0$

**for** $k = 0, 1, \ldots$ **do**
  $\boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}} \; f(\boldsymbol{u}) + \frac{\mu}{2}||\boldsymbol{A}\boldsymbol{u} - \boldsymbol{d}^k||_{l_2}^2$
  **if** stopping criterion is satisfied **then**
    break the loop
  **else**
    $\boldsymbol{d}^{k+1} = \boldsymbol{y} + (\boldsymbol{d}^k - \boldsymbol{A}\boldsymbol{u}^{k+1})$
  **end if**
**end for**

As it was shown in Yin et al. [2008] such constructed *Augmented Lagrangian* is equivalent to the *Bregman Iterative* method.

The augmented Lagrange function for problem (6.22) is

$$\mathcal{L}_A(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\lambda}, \mu) = f_1(\boldsymbol{u}) + f_2(\boldsymbol{v}) + \boldsymbol{\lambda}^T(\boldsymbol{T}\boldsymbol{u} - \boldsymbol{v}) + \frac{\mu}{2}||\boldsymbol{T}\boldsymbol{u} - \boldsymbol{v}||_{l_2}^2 \tag{6.28}$$

and can be seen as the Lagrangian approach for this problem with the quadratic penalty function

$$\begin{aligned} &\text{minimize}_{\boldsymbol{u},\boldsymbol{v}} && f_1(\boldsymbol{u}) + f_2(\boldsymbol{v}) + \tfrac{\mu}{2}||\boldsymbol{T}\boldsymbol{u} - \boldsymbol{v}||_{l_2}^2 \\ &\text{s.t.} && \boldsymbol{v} = \boldsymbol{T}\boldsymbol{u} \end{aligned}$$

Algorithm 11 can be extended to work with the *Variable Splitting* approach

**Algorithm 12** Variable Splitting Augmented Lagrangian

**Require:** $\mu > 0, \boldsymbol{v}^0, \boldsymbol{d}^0$

**for** $k = 0, 1, \ldots$ **do**
  $\boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}} f_1(\boldsymbol{u}) + \frac{\mu}{2}||\boldsymbol{T}\boldsymbol{u} - \boldsymbol{v}^k - \boldsymbol{d}^k||_{l_2}^2$
  $\boldsymbol{v}^{k+1} \in \arg\min_{\boldsymbol{v}} f_2(\boldsymbol{v}) + \frac{\mu}{2}||\boldsymbol{T}\boldsymbol{u}^{k+1} - \boldsymbol{v} - \boldsymbol{d}^k||_{l_2}^2$
  **if** stopping criterion is satisfied **then**
    break the loop
  **else**
    $\boldsymbol{d}^{k+1} = \boldsymbol{d}^k - \boldsymbol{T}\boldsymbol{u}^{k+1} + \boldsymbol{v}^{k+1}$
  **end if**
**end for**

Algorithm 12 also occurs under the name *Alternating-direction Method of Multipliers* and, in particular in the image processing community, as the *Alternating Split Bregman* algorithm [Combettes and Pesquet 2010]. Similar steps were also used to derive *SALSA* [Alfonso et al. 2009]. In the *Variable Splitting Augmented Lagrangian* algorithm the crucial step is computing both proximity operators $prox^T_{\frac{1}{\mu}f_1}(\boldsymbol{v}^k + \boldsymbol{d}^k)$ and $prox_{\frac{1}{\mu}f_2}(\boldsymbol{T}\boldsymbol{u}^{k+1} - \boldsymbol{d}^k)$ efficiently (see Chapter 5 for definitions of both proximity operators).

### 6.3.3 Variable Splitting Augmented Lagrangian Method

Let reformulate problem (3.7) as follows

$$\begin{aligned} &\text{minimize}_{\boldsymbol{u},\boldsymbol{s}_{(a)},\boldsymbol{s}_{(r)},\boldsymbol{s}_{(i)},\boldsymbol{r}} && \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2+\kappa_a||\boldsymbol{s}_{(a)}||_{l_1}+\kappa_r||\boldsymbol{s}_{(r)}||_{l_1}+\kappa_i||\boldsymbol{s}_{(i)}||_{l_1} \\ &\text{s.t.} && \boldsymbol{B}_a\boldsymbol{u}=\boldsymbol{s}_{(a)} \\ & && \boldsymbol{B}_r\boldsymbol{u}=\boldsymbol{s}_{(r)} \\ & && \boldsymbol{u}=(\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}+(\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)} \end{aligned} \tag{6.29}$$

where $\delta_1 = [1,0]^T$, $\delta_2 = [0,1]^T$. Put $\vec{\mu} := [\mu_a, \mu_r, \mu_i]$, $\boldsymbol{s} := \left[\boldsymbol{s}_{(a)}, \boldsymbol{s}_{(r)}, \boldsymbol{s}_{(i)}\right]$ and $\boldsymbol{d} := \left[\boldsymbol{d}_{(a)}, \boldsymbol{d}_{(r)}, \boldsymbol{d}_{(i)}\right]$. Applying the *Variable Splitting Augmented Lagrangian* method (Algorithm 12) to problem (6.29) yields the following algorithm

**Algorithm 13** Variable Splitting Augmented Lagrangian for Problem (6.29)

**Require:** $\vec{\mu} > 0, \boldsymbol{s}^0, \boldsymbol{d}^0, \boldsymbol{r}^0$

1: **for** $k = 0, 1, \ldots$ **do**

2: $$\begin{aligned}\boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}} \; & \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \\ & + \tfrac{\mu_a}{2}||\boldsymbol{B}_a\boldsymbol{u}-\boldsymbol{s}_{(a)}^k-\boldsymbol{d}_{(a)}^k||_{l_2}^2 \\ & + \tfrac{\mu_r}{2}||\boldsymbol{B}_r\boldsymbol{u}-\boldsymbol{s}_{(r)}^k-\boldsymbol{d}_{(r)}^k||_{l_2}^2 \\ & + \tfrac{\mu_i}{2}||\boldsymbol{u}-(\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}^k-(\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)}^k-\boldsymbol{d}_{(i)}^k||_{l_2}^2\end{aligned}$$

3: $\boldsymbol{s}_{(a)}^{k+1} \in \arg\min_{\boldsymbol{s}_{(a)}} \kappa_a||\boldsymbol{s}_{(a)}||_{l_1} + \frac{\mu_a}{2}||\boldsymbol{B}_a\boldsymbol{u}^{k+1}-\boldsymbol{s}_{(a)}-\boldsymbol{d}_{(a)}^k||_{l_2}^2$

4: $\boldsymbol{s}_{(r)}^{k+1} \in \arg\min_{\boldsymbol{s}_{(r)}} \kappa_r||\boldsymbol{s}_{(r)}||_{l_1} + \frac{\mu_r}{2}||\boldsymbol{B}_r\boldsymbol{u}^{k+1}-\boldsymbol{s}_{(r)}-\boldsymbol{d}_{(r)}^k||_{l_2}^2$

5: $\boldsymbol{s}_{(i)}^{k+1} \in \arg\min_{\boldsymbol{s}_{(i)}} \kappa_i||\boldsymbol{s}_{(i)}||_{l_1} + \frac{\mu_i}{2}||\boldsymbol{u}^{k+1}-(\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}^k-(\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)}-\boldsymbol{d}_{(i)}^k||_{l_2}^2$

6: $\boldsymbol{r}^{k+1} \in \arg\min_{\boldsymbol{r}} ||\boldsymbol{u}^{k+1}-(\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}-(\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)}^{k+1}-\boldsymbol{d}_{(i)}^k||_{l_2}^2$

7: $\boldsymbol{d}_{(a)}^{k+1} = \boldsymbol{d}_{(a)}^k - \boldsymbol{B}_a\boldsymbol{u}^{k+1} + \boldsymbol{s}_{(a)}^{k+1}$

8: $\boldsymbol{d}_{(r)}^{k+1} = \boldsymbol{d}_{(r)}^k - \boldsymbol{B}_r\boldsymbol{u}^{k+1} + \boldsymbol{s}_{(r)}^{k+1}$

9: $\boldsymbol{d}_{(i)}^{k+1} = \boldsymbol{d}_{(i)}^k - \boldsymbol{u}^{k+1} + (\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}^{k+1} + (\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)}^{k+1}$

10: **end for**

For each $j \in \{a, r\}$ the subproblem

$$\boldsymbol{s}_{(j)}^{k+1} \in \arg\min_{\boldsymbol{s}_{(j)}} \kappa_j||\boldsymbol{s}_{(j)}||_{l_1} + \frac{\mu_j}{2}||\boldsymbol{B}_j\boldsymbol{u}^{k+1}-\boldsymbol{s}_{(j)}-\boldsymbol{d}_{(j)}^k||_{l_2}^2 \tag{6.30}$$

can be recognized as the denoising problem and the solution can be obtained by applying the soft-thresholding operator (Definition 5.6; see also Theorem 5.1), that is

$$\boldsymbol{s}_{(j)}^{k+1} = S_{\frac{\kappa_j}{\mu_j}}(\boldsymbol{B}_j\boldsymbol{u}^{k+1}-\boldsymbol{d}_{(j)}^k) \tag{6.31}$$

Based on Corollary 5.1 we can derive the update step for $\boldsymbol{s}_{(i)}$

$$\begin{aligned}\boldsymbol{s}_{(i)}^{k+1} &= S_{\frac{\kappa_i}{\mu_i}}(\boldsymbol{s}_{(i)}-(\boldsymbol{I}\otimes\delta_2^T)((\boldsymbol{I}\otimes\delta_2)\boldsymbol{s}_{(i)}-(\boldsymbol{u}^{k+1}-(\boldsymbol{I}\otimes\delta_1)\boldsymbol{r}^k-\boldsymbol{d}_{(i)}^k))) \\ &= S_{\frac{\kappa_i}{\mu_i}}((\boldsymbol{I}\otimes\delta_2^T)(\boldsymbol{u}^{k+1}-\boldsymbol{d}_{(i)}^k))\end{aligned} \tag{6.32}$$

Note that $\boldsymbol{B}_i = (\boldsymbol{I}\otimes\delta_2^T)$ can be interpreted in the complex-valued vector space as a function which cuts off the real part of a given vector and leaves the imaginary part of the vector intact.

The next subproblem that has to be solved is

$$\boldsymbol{r}^{k+1} \in \arg\min_{\boldsymbol{r}} ||\boldsymbol{u}^{k+1} - (\boldsymbol{I} \otimes \delta_1)\boldsymbol{r} - (\boldsymbol{I} \otimes \delta_2)\boldsymbol{s}_{(i)}^{k+1} - \boldsymbol{d}_{(i)}^{k}||_{l_2}^2 \tag{6.33}$$

Taking derivative w.r.t. $\boldsymbol{r}$ yields the following update step for $\boldsymbol{r}$

$$\boldsymbol{r}^{k+1} = (\boldsymbol{I} \otimes \delta_1^T)(\boldsymbol{u}^{k+1} - \boldsymbol{d}_{(i)}^{k}) \tag{6.34}$$

The operator $(\boldsymbol{I} \otimes \delta_1^T)$ can be interpreted in the complex-valued vector space as a function which cuts off the imaginary part of a given vector and leaves the real part of the vector intact.

Finally, consider the following subproblem

$$\begin{aligned} \boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}} \quad & \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \\ & + \tfrac{\mu_a}{2}||\boldsymbol{B}_a\boldsymbol{u} - \boldsymbol{s}_{(a)}^{k} - \boldsymbol{d}_{(a)}^{k}||_{l_2}^2 \\ & + \tfrac{\mu_r}{2}||\boldsymbol{B}_r\boldsymbol{u} - \boldsymbol{s}_{(r)}^{k} - \boldsymbol{d}_{(r)}^{k}||_{l_2}^2 \\ & + \tfrac{\mu_i}{2}||\boldsymbol{u} - (\boldsymbol{I} \otimes \delta_1)\boldsymbol{r}^{k} - (\boldsymbol{I} \otimes \delta_2)\boldsymbol{s}_{(i)}^{k} - \boldsymbol{d}_{(i)}^{k}||_{l_2}^2 \end{aligned} \tag{6.35}$$

which can be identified as the elastic model (Definition 4.1 with operators: $\boldsymbol{B}_a$, $\boldsymbol{B}_r$ and $\boldsymbol{I}$) and the minimizer can be found by solving the following system of linear equations

$$(\boldsymbol{X}^H\boldsymbol{X} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{B}_r^T\boldsymbol{B}_r + \mu_i\boldsymbol{I})\boldsymbol{u} = \boldsymbol{b} \tag{6.36}$$

where

$$\begin{aligned} \boldsymbol{v}_{(a)} &:= \boldsymbol{s}_{(a)}^{k} + \boldsymbol{d}_{(a)}^{k} \\ \boldsymbol{v}_{(r)} &:= \boldsymbol{s}_{(r)}^{k} + \boldsymbol{d}_{(r)}^{k} \\ \boldsymbol{v}_{(i)} &:= (\boldsymbol{I} \otimes \delta_1)\boldsymbol{r}^{k} + (\boldsymbol{I} \otimes \delta_2)\boldsymbol{s}_{(i)}^{k} + \boldsymbol{d}_{(i)}^{k} \\ \boldsymbol{b} &:= \boldsymbol{X}^H\boldsymbol{y} + \mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{v}_{(i)} \end{aligned}$$

It was shown in Section 4.2 that if

$$\boldsymbol{A} := \boldsymbol{X}^H\boldsymbol{X} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{B}_r^T\boldsymbol{B}_r + \mu_i\boldsymbol{I}$$

is the elastic matrix and $\boldsymbol{b}$ is the elastic observation then the solution is

$$\boldsymbol{u} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\boldsymbol{F}\boldsymbol{b} \tag{6.37}$$

where

$$\hat{\boldsymbol{D}} := (\boldsymbol{I}_{j\in\mathcal{J}} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{D} + \mu_i\boldsymbol{I})^{-1}$$

and

$$\boldsymbol{D} := \boldsymbol{F}\boldsymbol{B}_r^T(\boldsymbol{F}\boldsymbol{B}_r^T)^T$$

Moreover, since $\boldsymbol{D}$ is a diagonal matrix (see eq. (4.11)), the matrix $\boldsymbol{I}_{j\in\mathcal{J}} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{D} + \mu_i\boldsymbol{I}$ is also diagonal with strictly positive entries. Therefore its inverse $\hat{\boldsymbol{D}}$ can be computed efficiently.

Similarly to eq. (4.12), the operator $\hat{\boldsymbol{D}}$ can be also seen as the following function in the Fourier domain

$$(\hat{\boldsymbol{D}}\boldsymbol{u})_{(\omega,\gamma)}^{\mathcal{M}} = (\delta_{\gamma\in\mathcal{J}} + \mu_a + \mu_r(|e^{-i2\pi\frac{\omega}{n_y}} - 1|^2 + |e^{-i2\pi\frac{\gamma}{n_x}} - 1|^2) + \mu_i)^{-1}\boldsymbol{u}_{(\omega,\gamma)}^{\mathcal{M}} \tag{6.38}$$

To reduce of the use of the Fourier transform $\boldsymbol{F}$ per iteration, we can pull $\boldsymbol{F}$ inside $\boldsymbol{b}$, that is

$$\begin{aligned}\boldsymbol{F}\boldsymbol{b} &= \boldsymbol{F}(\boldsymbol{X}^H\boldsymbol{y} + \mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{v}_{(i)}) \\ &= \boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{I}_{\cdot,\mathcal{J}}\boldsymbol{y} + \mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{v}_{(i)}) \\ &= \boldsymbol{I}_{\cdot,\mathcal{J}}\boldsymbol{y} + \boldsymbol{F}(\mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{v}_{(i)})\end{aligned} \tag{6.39}$$

To sum up the new version of Algorithm 13 is

**Algorithm 14** Variable Splitting Augmented Lagrangian for Problem (6.29) - Explicit Version

**Require:** $\vec{\mu} > 0, \boldsymbol{s}^0, \boldsymbol{d}^0, \boldsymbol{r}^0$

**for** $k = 0, 1, \ldots$ **do**
- $\boldsymbol{v}_{(a)} := \boldsymbol{s}^k_{(a)} + \boldsymbol{d}^k_{(a)}$
- $\boldsymbol{v}_{(r)} := \boldsymbol{s}^k_{(r)} + \boldsymbol{d}^k_{(r)}$
- $\boldsymbol{v}_{(i)} := (\boldsymbol{I} \otimes \delta_1)\boldsymbol{r}^k + (\boldsymbol{I} \otimes \delta_2)\boldsymbol{s}^k_{(i)} + \boldsymbol{d}^k_{(i)}$
- $\boldsymbol{u}^{k+1} = \boldsymbol{F}^H\hat{\boldsymbol{D}}(\boldsymbol{I}_{\cdot,\mathcal{J}}\boldsymbol{y} + \boldsymbol{F}(\mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{v}_{(i)}))$
- $\boldsymbol{s}^{k+1}_{(a)} = S_{\frac{\kappa_a}{\mu_a}}(\boldsymbol{B}_a\boldsymbol{u}^{k+1} - \boldsymbol{d}^k_{(a)})$
- $\boldsymbol{s}^{k+1}_{(r)} = S_{\frac{\kappa_r}{\mu_r}}(\boldsymbol{B}_r\boldsymbol{u}^{k+1} - \boldsymbol{d}^k_{(r)})$
- $\boldsymbol{s}^{k+1}_{(i)} = S_{\frac{\kappa_i}{\mu_i}}((\boldsymbol{I} \otimes \delta_2^T)(\boldsymbol{u}^{k+1} - \boldsymbol{d}^k_{(i)}))$
- $\boldsymbol{r}^{k+1} = (\boldsymbol{I} \otimes \delta_1^T)(\boldsymbol{u}^{k+1} - \boldsymbol{d}^k_{(i)})$
- $\boldsymbol{d}^{k+1}_{(a)} = \boldsymbol{d}^k_{(a)} - \boldsymbol{B}_a\boldsymbol{u}^{k+1} + \boldsymbol{s}^{k+1}_{(a)}$
- $\boldsymbol{d}^{k+1}_{(r)} = \boldsymbol{d}^k_{(r)} - \boldsymbol{B}_r\boldsymbol{u}^{k+1} + \boldsymbol{s}^{k+1}_{(r)}$
- $\boldsymbol{d}^{k+1}_{(i)} = \boldsymbol{d}^k_{(i)} - \boldsymbol{u}^{k+1} + (\boldsymbol{I} \otimes \delta_1)\boldsymbol{r}^{k+1} + (\boldsymbol{I} \otimes \delta_2)\boldsymbol{s}^{k+1}_{(i)}$

**end for**

Algorithm 14 requires only two invocations of the Fourier transform per iteration. In addition, it turns out that all subproblems from Algorithm 13 can be solved in closed form and so we may expect fast performance per iteration of Algorithm 14.

### 6.3.4 Alternative Splitting

We can consider alternative splitting to the one presented in problem (6.29)

$$\begin{aligned}&\text{minimize}_{\boldsymbol{u},\boldsymbol{s}_{(a)},\boldsymbol{s}_{(r)},\boldsymbol{s}_{(i)}} && \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \kappa_a||\boldsymbol{s}_{(a)}||_{l_1} + \kappa_r||\boldsymbol{s}_{(r)}||_{l_1} + \kappa_i||\boldsymbol{s}_{(i)}||_{l_1} \\ &\text{s.t.} && \boldsymbol{B}_a\boldsymbol{u} = \boldsymbol{s}_{(a)} \\ &&& \boldsymbol{B}_r\boldsymbol{u} = \boldsymbol{s}_{(r)} \\ &&& \boldsymbol{B}_i\boldsymbol{u} = \boldsymbol{s}_{(i)}\end{aligned} \tag{6.40}$$

One of the benefits of the new splitting scheme given by model (6.40) is that all constraints are treated in the same way. Therefore all subproblems

$$\forall_{j\in\{a,r,i\}} \quad \boldsymbol{s}^{k+1}_{(j)} \in \arg\min_{\boldsymbol{s}_{(j)}} \kappa_j||\boldsymbol{s}_{(j)}||_{l_1} + \frac{\mu_j}{2}||\boldsymbol{B}_j\boldsymbol{u}^{k+1} - \boldsymbol{s}_{(j)} - \boldsymbol{d}^k_{(j)}||_{l_2}^2 \tag{6.41}$$

can be solved by applying the soft-thresholding operator

$$\forall_{j\in\{a,r,i\}} \quad \boldsymbol{s}^{k+1}_{(j)} = S_{\frac{\kappa_j}{\mu_j}}(\boldsymbol{B}_j\boldsymbol{u}^{k+1} - \boldsymbol{d}^k_{(j)}) \tag{6.42}$$

In order to obtain the update step for $\boldsymbol{u}$ we have to solve the following problem

$$\begin{aligned}\boldsymbol{u}^{k+1} \in \arg\min_{\boldsymbol{u}} \quad & \tfrac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2 + \\ & + \tfrac{\mu_a}{2}||\boldsymbol{B}_a\boldsymbol{u}-\boldsymbol{s}_{(a)}^k-\boldsymbol{d}_{(a)}^k||_{l_2}^2 \\ & + \tfrac{\mu_r}{2}||\boldsymbol{B}_r\boldsymbol{u}-\boldsymbol{s}_{(r)}^k-\boldsymbol{d}_{(r)}^k||_{l_2}^2 \\ & + \tfrac{\mu_i}{2}||\boldsymbol{B}_i\boldsymbol{u}-\boldsymbol{s}_{(i)}^k-\boldsymbol{d}_{(i)}^k||_{l_2}^2\end{aligned} \tag{6.43}$$

which is equivalent to solving the following elastic linear system

$$(\boldsymbol{X}^H\boldsymbol{X} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{B}_r^T\boldsymbol{B}_r + \mu_i\boldsymbol{B}_i^T\boldsymbol{B}_i)\boldsymbol{u} = \boldsymbol{b} \tag{6.44}$$

where

$$\begin{aligned}\boldsymbol{v}_{(a)} &:= \boldsymbol{s}_{(a)}^k + \boldsymbol{d}_{(a)}^k \\ \boldsymbol{v}_{(r)} &:= \boldsymbol{s}_{(r)}^k + \boldsymbol{d}_{(r)}^k \\ \boldsymbol{v}_{(i)} &:= \boldsymbol{s}_{(i)}^k + \boldsymbol{d}_{(r)}^k \\ \boldsymbol{b} &:= \boldsymbol{X}^H\boldsymbol{y} + \mu_a\boldsymbol{B}_a^T\boldsymbol{v}_{(a)} + \mu_r\boldsymbol{B}_r^T\boldsymbol{v}_{(r)} + \mu_i\boldsymbol{B}_i^T\boldsymbol{v}_{(i)}\end{aligned}$$

It was shown in Section 4.2 that the solution of the system given by eq. (6.44) is

$$\boldsymbol{u} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\boldsymbol{F}\boldsymbol{b} \tag{6.45}$$

where

$$\hat{\boldsymbol{D}} := (\boldsymbol{I}_{j\in\mathcal{J}} + \mu_a\boldsymbol{I} + \mu_r\boldsymbol{D} + \mu_i\boldsymbol{D}_i)^{-1}$$

and

$$\begin{aligned}\boldsymbol{D}_i &:= \boldsymbol{F}\boldsymbol{B}_i^T\boldsymbol{B}_i\boldsymbol{F}^H \\ \boldsymbol{D} &:= \boldsymbol{F}\boldsymbol{B}_r^T(\boldsymbol{F}\boldsymbol{B}_r^T)^T\end{aligned}$$

Therefore we can conclude that the update step for $\boldsymbol{u}$ should be

$$\boldsymbol{u}^{k+1} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\boldsymbol{F}\boldsymbol{b} \tag{6.46}$$

Section 4.2 provides details how the solution can be obtained from eq. (6.45).

The update steps for $\boldsymbol{u}$ and $\boldsymbol{s}_{(j)}$ for every $j \in \{a, r, i\}$ yield the following algorithm

**Algorithm 15** Variable Splitting Augmented Lagrangian for Problem (6.40)

**Require:** $\vec{\mu} > 0, \boldsymbol{s}^0, \boldsymbol{d}^0$

1: **for** $k = 0, 1, \ldots$ **do**
2: For every $j \in \{a, r, i\}$ compute $\boldsymbol{v}_{(j)} := \boldsymbol{s}_{(j)}^k + \boldsymbol{d}_{(j)}^k$
3: $\boldsymbol{u}^{k+1} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\left(\boldsymbol{I}_{\cdot,\mathcal{J}}\,\boldsymbol{y} + \boldsymbol{F}\left(\sum_{j\in\{a,r,i\}}\mu_j\boldsymbol{B}_j^T\boldsymbol{v}_{(j)}\right)\right)$
4: For every $j \in \{a, r, i\}$ compute the update step $\boldsymbol{s}_{(j)}^{k+1} = S_{\frac{\kappa_j}{\mu_j}}(\boldsymbol{B}_j\boldsymbol{u}^{k+1} - \boldsymbol{d}_{(j)}^k)$
5: For every $j \in \{a, r, i\}$ compute the update step $\boldsymbol{d}_{(j)}^{k+1} = \boldsymbol{d}_{(j)}^k - \boldsymbol{B}_j\boldsymbol{u}^{k+1} + \boldsymbol{s}_{(j)}^{k+1}$
6: **end for**

This algorithm, similarly to Algorithm 14, requires only two invocations of the Fourier transform per iteration. However, unlike Algorithm 14, the update steps for $\boldsymbol{v}_{(j)}$, $\boldsymbol{s}_{(j)}$ and $\boldsymbol{d}_{(j)}$ for every $j \in \{a, r, i\}$ are treated equally. Therefore Algorithm 15 can be easily extended to work with more than three penalizing operators providing that the operator $\hat{\boldsymbol{D}}$ or the matrix-vector multiplication $\hat{\boldsymbol{D}}\boldsymbol{v}$ is specified.

Chapter 7

# Second Order Method

## Contents



In this chapter, the MRI energy function (Definition 3.11) is reformulated by adding additional $l_2$-norm terms. The new energy function, called in this thesis the MRI elastic energy function, that arises from the reformulation opens the door for second order methods such as *Interior Point* method.

Section 7.1 defines the new energy function where the regularization term contains both $l_1$-norm terms and $l_2$-norm terms. The dualization of the new problem, which is a working space for new algorithms, is also shown. . If we want to solve the dual problem, we have to know how to compute the inverse of the so called elastic matrix $\boldsymbol{A}$. However, from the algorithmic perspective it is suffice to show how to compute $\boldsymbol{A}^{-1}\boldsymbol{v}$ instead of showing the whole inverse $\boldsymbol{A}^{-1}$. Section 7.2 shows how to do this efficiently. Section 7.3 introduces a first order method which works on the dual problem. This method is strongly based on the *Fast Gradient Projection* algorithm introduced in Subsection 6.2.1. Section 7.4 shows how to use the second order method called the *Interior Point* method in order to solve the problem of finding the minimizer of the MRI elastic energy function.

## 7.1 MRI Elastic Energy Function

Consider the following extension to the original model

**Definition 7.1** (MRI Elastic Energy Function)**.** *Let $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$, $\boldsymbol{X}$ be, respectively, the wavelet transform, the finite difference operator, the imaginary part penalizing operator and the subsampling Fourier operator, and assume that all of them are tailored to work with the real-valued surrogate of the complex set $\mathcal{C} := \mathbb{R}^2$. For every $\boldsymbol{u} \in \mathcal{C}^n$ let*

$$\Psi^{MRI}_{Elastic}(\boldsymbol{u}) := \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||^2_{l_2} + \sum_{j\in\{a,r,i\}} \left\{\kappa_j||\boldsymbol{B}_j\boldsymbol{u}||_{l_1} + \frac{\mu_j}{2}||\boldsymbol{B}_j\boldsymbol{u}||^2_{l_2}\right\} \tag{7.1}$$

*where for each* $j \in \{a, r, i\}$ $\kappa_j := \tau_j \sigma$, $\tau_j$ *is a model parameter,* $\mu_j$ *is an elastic parameter and* $\sigma^2$ *is the noise variance. The function* $\Psi^{MRI}_{Elastic}(\cdot)$ *is called the MRI elastic energy function. Operators* $\boldsymbol{B}_a$, $\boldsymbol{B}_r$, $\boldsymbol{B}_i$ *are also called penalizing operators. Then the extended model is*

$$\underset{\boldsymbol{u}}{\text{minimize}} \ \Psi^{MRI}_{Elastic}(\boldsymbol{u}) \tag{7.2}$$

In this chapter we will work with the special case of the elastic matrix and the elastic observation. Both are defined as follows

**Definition 7.2** (Elastic Matrix and Elastic Observation)**.** *Put*

$$\boldsymbol{v} := \sum_{j \in \{a,r,i\}} \kappa_j \boldsymbol{B}_j^T \boldsymbol{p}_{(j)} = \boldsymbol{B}^T \boldsymbol{p}$$

*Let*

$$\boldsymbol{A} := \boldsymbol{X}^H \boldsymbol{X} + \sum_{j \in \{a,r,i\}} \mu_j \boldsymbol{B}_j^T \boldsymbol{B}_j \tag{7.3}$$

*be the elastic matrix and*

$$\boldsymbol{b} := \boldsymbol{X}^H \boldsymbol{y} - \boldsymbol{v} \tag{7.4}$$

*be the elastic observation. Note that, the elastic matrix is symmetric in the sense described in Section 3.4.*

The following optimization problem

$$\underset{\boldsymbol{u}}{\text{minimize}} \, \Psi^{\text{MRI}}_{\text{Elastic}}(\boldsymbol{u})$$

can be transformed into[1]

$$\underset{\boldsymbol{u} \in \mathcal{P}}{\text{minimize}} \frac{1}{2} ||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||^2_{l_2} + \sum_{j \in \{a,r,i\}} \left\{ \kappa_j ||\boldsymbol{B}_j \boldsymbol{u}||_{l_1} + \frac{\mu_j}{2} ||\boldsymbol{B}_j \boldsymbol{u}||^2_{l_2} \right\} \tag{7.5}$$

$$= \underset{\boldsymbol{p} \in \mathcal{P}}{\text{maximize}} \, \underset{\boldsymbol{u}}{\text{minimize}} \frac{1}{2} ||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||^2_{l_2} + \boldsymbol{v}^T \boldsymbol{u} + \frac{1}{2} \sum_{j \in \{a,r,i\}} \mu_j ||\boldsymbol{B}_j \boldsymbol{u}||^2_{l_2} \tag{7.6}$$

where $\boldsymbol{v}^T := \sum_j \kappa_j \boldsymbol{p}^T_{(j)} \boldsymbol{B}_j$ and $P$ is the P-Set (Definition 6.2). Note that, problem (7.6) consists of the inner optimization problem

$$\Psi^{\text{MRI}}_{\text{Inner}}(\boldsymbol{p}) := \underset{\boldsymbol{u}}{\text{minimize}} \frac{1}{2} ||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||^2_{l_2} + \boldsymbol{v}^T \boldsymbol{u} + \frac{1}{2} \sum_j \mu_j ||\boldsymbol{B}_j \boldsymbol{u}||^2_{l_2}$$

and the outer optimization problem

$$\underset{\boldsymbol{p} \in \mathcal{P}}{\text{maximize}} \, \Psi^{\text{MRI}}_{\text{Inner}}(\boldsymbol{p})$$

Lemma 7.1 shows that the objective function of the optimization problem (7.6) has the quadratic form

[1] Similar transformation was presented in more details in Subsection 6.2.7.

**Lemma 7.1.** *Let* $q(\boldsymbol{u}; \bar{\boldsymbol{b}}, \bar{\boldsymbol{A}}) := \frac{1}{2}\boldsymbol{u}^T\bar{\boldsymbol{A}}\boldsymbol{u} - \bar{\boldsymbol{b}}^T\boldsymbol{u}$. *Then, for every* $\boldsymbol{u}$, *we have*

$$\frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \boldsymbol{v}^T\boldsymbol{u} + \frac{1}{2}\sum_j \mu_j||\boldsymbol{B}_j\boldsymbol{u}||_{l_2}^2 = q(\boldsymbol{u}; \boldsymbol{b}, \boldsymbol{A}) + C \tag{7.7}$$

*where* $\boldsymbol{A}$ *is the elastic matrix,* $\boldsymbol{b}$ *is the elastic observation and* $C$ *is some expression independent of* $\boldsymbol{u}$ *and* $\boldsymbol{p}$.

*Proof.*

$$\begin{aligned}
&\frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \boldsymbol{v}^T\boldsymbol{u} + \frac{1}{2}\sum_j \mu_j||\boldsymbol{B}_j\boldsymbol{u}||_{l_2}^2 \\
&= \frac{1}{2}(\boldsymbol{u}^T\boldsymbol{X}^H - \boldsymbol{y}^T)(\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}) + \boldsymbol{v}^T\boldsymbol{u} + \frac{1}{2}\sum_j \mu_j\boldsymbol{u}^T\boldsymbol{B}_j^T\boldsymbol{B}\boldsymbol{u} \\
&= \frac{1}{2}\boldsymbol{u}^T(\boldsymbol{X}^H\boldsymbol{X} + \sum_j \mu_j\boldsymbol{B}_j^T\boldsymbol{B}_j)\boldsymbol{u} + (\boldsymbol{v}^T - \boldsymbol{y}^T\boldsymbol{X})\boldsymbol{u} + C \\
&= \frac{1}{2}\boldsymbol{u}^T\boldsymbol{A}\boldsymbol{u} - \boldsymbol{b}^T\boldsymbol{u} + C = q(\boldsymbol{u}; \boldsymbol{b}, \boldsymbol{A}) + C
\end{aligned}$$

□

The next lemma shows that the solution of the inner optimization problem $\Psi_{\text{Inner}}^{\text{MRI}}(\boldsymbol{p})$ is $\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{b}$

**Lemma 7.2.** *For every fixed* $\boldsymbol{p}$, *consider the following problem*

$$\underset{\boldsymbol{u}}{\text{minimize}}\, \frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 + \boldsymbol{v}^T\boldsymbol{u} + \frac{1}{2}\sum_j ||\boldsymbol{B}_j\boldsymbol{u}||^2 \tag{7.8}$$

*Then the solution of problem (7.8) is*

$$\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{7.9}$$

*where* $\boldsymbol{A}$ *is the elastic matrix and* $\boldsymbol{b}$ *is the elastic observation.*

*Proof.* We can obtain the solution of the problem (7.8) by solving the elastic linear system given by eq. (4.2). It is easy to see that the solution of this system is $\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{b}$. □

Interestingly, when we apply $q(\boldsymbol{u}; \boldsymbol{b}, \boldsymbol{A})$ to the argument $\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{b}$ we obtain another quadratic function. Thus the inner optimization problem $\Psi_{\text{Inner}}^{\text{MRI}}(\cdot)$ has a quadratic form given $\boldsymbol{u}^\star$. This is shown by the next lemma

**Lemma 7.3.** *Assume that* $\boldsymbol{A}$ *is the elastic matrix,* $\boldsymbol{b}$ *is the elastic observation and let* $\boldsymbol{u}^\star := \boldsymbol{A}^{-1}\boldsymbol{b}$. *Then*

$$q(\boldsymbol{u}^\star; \boldsymbol{b}, \boldsymbol{A}) = -\frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b} \tag{7.10}$$

*Proof.*

$$\begin{aligned}
q(\boldsymbol{u}^\star; \boldsymbol{b}, \boldsymbol{A}) &= \frac{1}{2}(\boldsymbol{u}^\star)^T\boldsymbol{A}\boldsymbol{u}^\star - \boldsymbol{b}^T\boldsymbol{u}^\star \\
&= \frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b} = -\frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}
\end{aligned}$$

□

Therefore instead of dealing with problem (7.5) we can work with its dual which objective has the quadratic form given by eq. (7.10). Therefore we can conclude that

**Corollary 7.1.** *Let $\boldsymbol{A}$ be the elastic matrix and $\boldsymbol{b}$ the elastic observation. Then problem (7.5) is equivalent to*

$$-\underset{\boldsymbol{p}\in\mathcal{P}}{\text{minimize}}\left\{\Psi_{Inner}^{MRI}(\boldsymbol{p}) := \frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}\right\} \tag{7.11}$$

*and the solution $\boldsymbol{u}^\star$ can be obtained by applying $\boldsymbol{A}^{-1}$ to $\boldsymbol{b}$.*

By expanding $\boldsymbol{b}$ and canceling all terms that are independent of $\boldsymbol{p}$, we can work with the following optimization problem

$$\arg\min_{\boldsymbol{p}\in\mathcal{P}}\left\{\Psi_{\text{Inner}}^{\text{MRI}}(\boldsymbol{p}) := \frac{1}{2}\boldsymbol{p}^T\boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T\boldsymbol{p} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{B}^T\boldsymbol{p}\right\} \tag{7.12}$$

Therefore, in order to solve the optimization problem

$$\boldsymbol{u}^\star = \arg\min_{u}\Psi_{\text{Elastic}}^{\text{MRI}}(\boldsymbol{u}) \tag{7.13}$$

we can solve the following quadratically constrained quadratic program

$$\boldsymbol{p}^\star = \arg\min_{\boldsymbol{p}\in\mathcal{P}}\Psi_{\text{Inner}}^{\text{MRI}}(\boldsymbol{p}) \tag{7.14}$$

and apply transformation $\boldsymbol{A}^{-1}$ to the argument $\boldsymbol{p}^\star$ in order to obtain the minimizer $\boldsymbol{u}^\star$, shortly $\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{p}^\star$.

Let $\boldsymbol{Q}$ be a symmetric and positive semidefinite matrix. In this thesis, every problem which has the following form

$$\begin{array}{ll}\arg\min_{\boldsymbol{p}} & \boldsymbol{p}^T\boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r}^T\boldsymbol{p} \\ \text{subject to} & ||\boldsymbol{p}_i||_{l_2} \leq 1\end{array} \tag{7.15}$$

is called the normed constrained quadratic program. Under this convention, model (7.14) can be seen as an instance of the normed constrained quadratic program.

## 7.2 Inverse of the Elastic Matrix

Since the optimization problem (7.12) requires the inverse of the elastic matrix, we have to find efficient way to compute $\boldsymbol{A}^{-1}$. Fortunately, we don't have to specify the inverse of the elastic matrix explicitly. Instead, we can show how to solve the elastic linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \tag{7.16}$$

where

$$\boldsymbol{A} = \boldsymbol{X}^H\boldsymbol{X} + \sum_{j\in\{a,r,i\}}\mu_j\boldsymbol{B}_j^T\boldsymbol{B}_j$$

and

$$\boldsymbol{b} = \sum_{j\in\{a,r,i\}}\kappa_j\boldsymbol{B}_j^T\boldsymbol{p}_{(j)}$$

Section 4.2 showed how to efficiently compute the solution of the system given by eq. (7.16). Hence the reading of Chapter 4 in order to understand the method of solving this kind of linear systems is vital.

## 7.3 Normed Constrained Quadratic FGP

Consider the following normed constrained quadratic program

$$\begin{array}{ll} \arg\min_{\boldsymbol{p}} & \frac{1}{2}\boldsymbol{p}^T\boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r}^T\boldsymbol{p} \\ \text{subject to} & ||\boldsymbol{p}_i||_{l_2} \leq 1 \end{array} \tag{7.17}$$

where $\boldsymbol{p}_i \in \mathcal{C} := \mathbb{R}^2$ and it is assumed that $\boldsymbol{Q}$ is a symmetric, positive semidefinite matrix. For $\boldsymbol{Q} := \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$ and $\boldsymbol{r}^T := -\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{B}^T$ problem (7.17) becomes equivalent to problem (7.14).

The *Fast Gradient Projection* (FGP) algorithm, introduced in Subsection 6.2.1, can be adapted in order to solve the problem (7.17). Recall that fixed point iteration of the FGP is

$$\begin{aligned} \boldsymbol{p}^k &= P_{\mathcal{P}}(\boldsymbol{q}^k - \alpha\nabla f(\boldsymbol{q}^k)) \\ t_{k+1} &= \frac{1+\sqrt{1+4\,t_k^2}}{2} \\ \boldsymbol{q}^{k+1} &= \boldsymbol{p}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{p}^k - \boldsymbol{p}^{k-1}) \end{aligned}$$

where $P_{\mathcal{P}}$ is a projection onto some set $\mathcal{P}$ and $f$ is the objective function. In the context of problem (7.17) the set $\mathcal{P}$ is the P-set (Definition 6.2) and $f(\boldsymbol{p}) := \boldsymbol{p}^T\boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r}^T\boldsymbol{p}$. Since the gradient of the objective function is $\nabla f(\boldsymbol{p}) = \boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r}$ we have the following algorithm

**Algorithm 16** Normed Constrained Quadratic Fast Gradient Projection

**Require:** $\boldsymbol{p}^0$, $\boldsymbol{q}^1$ , $t_1$ (by default $t_1 := 1$), $\alpha$

1: **for** $k = 1, 2, \ldots$ **do**
2: $\quad\boldsymbol{p}^k = P_{\mathcal{P}}(\boldsymbol{q}^k - \alpha\left(\boldsymbol{Q}\boldsymbol{q}^k + \boldsymbol{r}\right))$
3: $\quad$**if** stopping criterion is satisfied **then**
4: $\quad\quad$break the loop
5: $\quad$**else**
6: $\quad\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
7: $\quad\quad \boldsymbol{q}^{k+1} = \boldsymbol{p}^k + (\frac{t_k-1}{t_{k+1}})(\boldsymbol{p}^k - \boldsymbol{p}^{k-1})$
8: $\quad$**end if**
9: **end for**

In the context of the problem

$$\arg\min_{\boldsymbol{u}} \ \Psi^{\text{MRI}}_{\text{Elastic}}(\boldsymbol{u})$$

we have $\boldsymbol{Q} := \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$ and so Algorithm 16 requires two invocations of the Fourier transform per iteration. Moreover, since equation $\boldsymbol{A}^{-1} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\boldsymbol{F}$ holds[2], one invocation of the Fourier transform is needed in order to compute $\boldsymbol{r}^T := -\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{B}^T = -\boldsymbol{y}^T\boldsymbol{I}_{\mathcal{J},\cdot}\hat{\boldsymbol{D}}\boldsymbol{F}\boldsymbol{B}^T$ and two additional in order to compute $\boldsymbol{u}^\star = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{F}^H\hat{\boldsymbol{D}}\left(\boldsymbol{I}_{\cdot,\mathcal{J}}\boldsymbol{y} - \boldsymbol{F}\boldsymbol{v}\right)$ where $\boldsymbol{b}$ is the elastic observation and $\boldsymbol{A}$ is the elastic matrix (Definition 7.2). However, since $\boldsymbol{r}$ can be computed in the preprocessing step and $\boldsymbol{u}^\star$ in the postprocessing step, the invocations of

[2] See eq. (4.10) in Section 4.2.

the Fourier transform required in computation of $\boldsymbol{r}$ and $\boldsymbol{u}^\star$ can be amortized over the whole performance of the algorithm. Naturally, if a backtracking line-search, for instance Algorithm 23, is used then the number of the Fourier transform invocations may increase.

## 7.4 Interior Point Method

### 7.4.1 Introduction

**Newton Method.** Consider the following unconstrained optimization problem

$$\underset{\boldsymbol{u}}{\text{minimize}} \; f(\boldsymbol{u}) \tag{7.18}$$

where $f$ is assumed to be convex and twice continuously differentiable function. In order to solve problem (7.18) first order methods such as *Steepest Descent* or second order methods can be used. The former methods use only information about the gradient to solve the optimization problem, whereas latter methods use also information about the Hessian in order to solve the problem.

*Newton* method is a popular second order method that can be used to solve the unconstrained problem (7.18). Algorithm 17 presents this method.

**Algorithm 17** Newton

**Require:** $\boldsymbol{u}^0$

1: **for** $k = 0, 1, \ldots$ **do**
2: Find direction $\boldsymbol{d}^k$ which satisfies $\nabla^2 f(\boldsymbol{u}^k)\boldsymbol{d}^k = -\nabla f(\boldsymbol{u}^k)$
3: $\alpha_k \in \arg\min_{\alpha>0} f(\boldsymbol{u}^k + \alpha \boldsymbol{d}^k)$
4: $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha_k \boldsymbol{d}^k$
5: **if** stopping criterion is satisfied **then**
6: break the loop
7: **end if**
8: **end for**

Algorithm 17 has to solve the following linear system

$$\nabla^2 f(\boldsymbol{u}^k)\boldsymbol{d}^k = -\nabla f(\boldsymbol{u}^k) \tag{7.19}$$

in order to find direction $\boldsymbol{d}^k$. Although sometimes the inverse of the Hessian $\nabla^2 f$ is known, quite often it is not the case. Then we have to resort to methods that compute the solution of the linear system given by eq. (7.19), for instance the method of *Gaussian elimination* or the *Conjugate Gradient* algorithm. The latter has this advantage over the former that iterations can be truncated in order to gain time and still obtain good approximation of the solution. Such the *Newton* method which employs *Conjugate Gradient* in order to find direction $\boldsymbol{d}^k$ is also called the *Truncated Newton* method.

Apart from the direction-search problem, the line-search $\alpha_k \in \arg\min_{\alpha>0} f(\boldsymbol{u}^k + \alpha \boldsymbol{d}^k)$ is also required. There are a few methods to find $\alpha_k$. In some circumstances the step-size $\alpha_k$ can be found analytically, in some others a line-search algorithm should be used (see Appendix G for some line-search methods).

**Interior Point Method.** Consider the following constrained convex optimization problem

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{u}} & f(\boldsymbol{u}) \\ \text{subject to} & g_i(\boldsymbol{u}) \leq 0 \quad i \in \{1, 2, \ldots, m\} \end{array} \tag{7.20}$$

where $f, g_1, g_2, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex and twice continuously differentiable functions. In addition, it is assumed that the problem is strictly feasible, that is, there exists $\boldsymbol{u} \in \text{dom}(f)$ such that $g_i(\boldsymbol{u}) < 0$ for all $i \in \{1, 2, \ldots, m\}$. Let $\mathcal{F}$ be the feasible set, that is $\mathcal{F} := \left\{\boldsymbol{u} \mid \forall_{j \in \{1,2,\ldots,m\}} g_j(\boldsymbol{u}) \leq 0\right\}$. Intuitively, the *Interior Point* method approaches a solution from the interior of the feasible set $\mathcal{F}$.

The constrained optimization problem (7.20) can be transformed into the unconstrained one by putting inequality constraints into the objective

$$\underset{\boldsymbol{u}}{\text{minimize}} \; f(\boldsymbol{u}) + \sum_{j=1}^{m} I_-(g_j(\boldsymbol{u})) \tag{7.21}$$

where $I_-$ is the indicator function defined as

$$I_-(a) = \begin{cases} 0 & \text{if } a \leq 0 \\ \infty & \text{if } a > 0 \end{cases}$$

The role of the indicator function $I_-(\cdot)$ is to prevent the solution from leaving the feasible set. For this reason the function $I_-(\cdot)$ is also called the barrier function.

In practice, however, another barrier function is often used, the so called logarithmic barrier function defined as

$$\hat{I}_-(a; t) := \begin{cases} -\frac{1}{t}\log(-a) & \text{if } a < 0 \\ \infty & \text{if } a \geq 0 \end{cases} \tag{7.22}$$

The logarithmic barrier function $\hat{I}_-(\cdot, t_k)$ can be seen as an approximation of the barrier function $I_-$ for the increasing sequence $(t_k)_{k \in \{0,1,2,\ldots\}}$. Figure 7.1 shows how $\hat{I}_-(\cdot, t_k)$ is changing and is approaching $I_-$ with respect to $t_k$. Since $\hat{I}_-$ is twice continuously differentiable and convex, increasing function in $a$ the whole objective $f(\boldsymbol{u}) + \sum_{j=1}^{m} \hat{I}_-(g_i(\boldsymbol{u}); t_k)$ is also twice differentiable and convex function. Thus, for fixed $t$, a second order method can be employed to solve the centering problem

$$\arg\min_{\boldsymbol{u}} \left\{ \Psi(\boldsymbol{u}; t) := f(\boldsymbol{u}) + \sum_{j=1}^{m} \hat{I}_-(g_j(\boldsymbol{u}); t) \right\} \tag{7.23}$$

Function $\Psi$ is called the centering function.

For fixed $t$, the gradient and the Hessian of the logarithmic barrier function $\hat{I}_-(\cdot; t)$ are given by [Boyd and Vandenberghe 2004]

$$\nabla \hat{I}_-(g_j(\boldsymbol{u}); t) = \frac{1}{-t\, g_j(\boldsymbol{u})} \nabla g_j(\boldsymbol{u}) \tag{7.24}$$

$$\nabla^2 \hat{I}_-(g_j(\boldsymbol{u}); t) = \frac{1}{t\, g_j(\boldsymbol{u})^2} \nabla g_j(\boldsymbol{u}) \nabla g_j(\boldsymbol{u})^T + \frac{1}{-t\, g_j(\boldsymbol{u})} \nabla^2 g_j(\boldsymbol{u}) \tag{7.25}$$

The whole algorithm can be summarized as follows

**Algorithm 18** Interior Point Method

**Require:** $\boldsymbol{u}^0$ such that for all $j \in \{1, 2, \dots, m\}$ $g_j(\boldsymbol{u}^0) < 0$ holds, $t_0 > 0$

**for** $k = 0, 1, \dots$ **do**
  Solve the centering problem $\hat{\boldsymbol{u}} \in \arg\min_{\boldsymbol{u}}$ $f(\boldsymbol{u}) + \sum_{j=1}^{m} \hat{I}_-(g_j(\boldsymbol{u}); t_k)$
  $t_{k+1} := \text{Update}(t_k)$
  **if** stopping criterion is satisfied **then**
    break the loop
  **end if**
**end for**

If the *Truncated Newton* method is used to solve the centering problem, Algorithm 18 can be written as follows (now the update step for $t_{k+1}$ depends on the step-size $\alpha_k$ as well)

**Algorithm 19** Truncated Newton Interior Point Method

**Require:** $\boldsymbol{u}^0$ such that for all $j \in \{1, 2, \dots, m\}$ $g_j(\boldsymbol{u}^0) < 0$ holds, $t_0 > 0$

**for** $k = 0, 1, \dots$ **do**
  Use an iterative solver to compute direction $\boldsymbol{d}^k$ which satisfies

$$\nabla^2 \Psi(\boldsymbol{u}^k; t_k) \boldsymbol{d}^k = -\nabla \Psi(\boldsymbol{u}^k; t_k)$$

  $\alpha_k \in \arg\min_{\alpha > 0} \Psi(\boldsymbol{u}^k + \alpha \boldsymbol{d}^k; t_k)$
  $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha_k \boldsymbol{d}^k$
  $t_{k+1} := \text{Update}(t_k, \alpha_k)$
  **if** stopping criterion is satisfied **then**
    break the loop
  **end if**
**end for**

Let assume that the dual of problem (7.20) is known, that is

$$\underset{\boldsymbol{v} \in \mathcal{D}}{\text{maximize}} \; d(\boldsymbol{v}) \tag{7.26}$$

for some feasible set $\mathcal{D}$ and a dual function $d$. By weak duality for every feasible point $\bar{\boldsymbol{v}} \in \mathcal{D}$ the inequality $d(\bar{\boldsymbol{v}}) \leq p^\star$ holds, where $p^\star$ is the optimal value of the primal problem (7.20). Providing that a dual point $\bar{\boldsymbol{v}}$ can be calculated, we can exploit information about the duality gap $\nu := f(\boldsymbol{u}) - d(\bar{\boldsymbol{v}})$ to develop the following modified version of Algorithm 19

**Algorithm 20** Truncated Newton Primal Dual Interior Point Method

**Require:** $\boldsymbol{u}^0$ such that for all $j \in \{1, 2, \ldots, m\}$ $g_j(\boldsymbol{u}^0) < 0$ holds, $t_0 > 0$

1: **for** $k = 0, 1, \ldots$ **do**
2: Use an iterative solver to compute direction $\boldsymbol{d}^k$ which satisfies

$$\nabla^2 \Psi(\boldsymbol{u}^k; t_k)\boldsymbol{d}^k = -\nabla \Psi(\boldsymbol{u}^k; t_k) \tag{7.27}$$

and where the maximal number of iterations required for solving the linear system (7.27) is computed based on the duality gap $\nu_k$
3: $\alpha_k \in \arg\min_{\alpha > 0} \Psi(\boldsymbol{u}^k + \alpha \boldsymbol{d}^k; t_k)$
4: $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha_k \boldsymbol{d}^k$
5: Construct dual point $\bar{\boldsymbol{v}}$
6: Evaluate duality gap $\nu_{k+1} := f(\boldsymbol{u}^{k+1}) - d(\bar{\boldsymbol{v}})$
7: $t_{k+1} := \text{Update}(t_k, \nu_{k+1}, \alpha_k)$
8: **if** stopping criterion is satisfied **then**
9: break the loop
10: **end if**
11: **end for**

Similar approach was also used in Kim et al. [2007] to develop the *Interior Point* method suitable for solving large-scale $l_1$-regularized least squares problems. The main difference between Algorithm 19 and Algorithm 20 is that in the latter the update step for $t$ and the maximal number of iterations required in order to find a direction $\boldsymbol{d}$ are explicitly based on the duality gap.

### 7.4.2 Application to the Normed Constrained Quadratic Program

Recall the normed constrained quadratic program (problem (7.17)) which can be equivalently written as

$$\begin{array}{ll} \arg\min_{\boldsymbol{p}} & \left\{ f(\boldsymbol{p}) := \frac{1}{2}\boldsymbol{p}^T \boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r}^T \boldsymbol{p} \right\} \\ \text{subject to} & ||\boldsymbol{p}_j||_{l_2}^2 - 1 \leq 0 \end{array} \tag{7.28}$$

where $\boldsymbol{p}_j \in \mathcal{C}$ and it is assumed that $\boldsymbol{Q}$ is a symmetric, positive semidefinite matrix. Moreover, the objective function $f$ and constraint's functions $g_j(\boldsymbol{p}) := ||\boldsymbol{p}_j||_{l_2}^2 - 1$ are twice continuously differentiable. In order to employ the *Truncated Newton Primal Dual Interior Point* method we have to specify, for fixed $t$, the gradient and the Hessian of the centering function

$$\Psi(\boldsymbol{p}; t) := f(\boldsymbol{p}) + \sum_{j=1}^{m} \hat{I}_-(g_j(\boldsymbol{p}); t)$$

where

$$\hat{I}_-(g_j(\boldsymbol{p}); t) = \begin{cases} -\frac{1}{t}\log(1 - ||\boldsymbol{p}_j||_{l_2}^2) & \text{if } ||\boldsymbol{p}_j||_{l_2}^2 < 1 \\ \infty & \text{otherwise} \end{cases} \tag{7.29}$$

The gradient and the Hessian of the objective function $f$ are just

$$\nabla f(\boldsymbol{p}) = \boldsymbol{Q}\boldsymbol{p} + \boldsymbol{r} \tag{7.30}$$

$$\nabla^2 f(\boldsymbol{p}) = \boldsymbol{Q} \tag{7.31}$$

Let $\boldsymbol{e}_j$ be the $j$-th standard vector in $\mathcal{C}^n$ [3]. Then the gradient of the logarithmic barrier function $\hat{I}_-(g_j(\cdot);t)$ can be computed to be

$$\nabla \hat{I}_-(g_j(\boldsymbol{p});t) = \frac{2}{t\left(1 - ||\boldsymbol{p}_j||_{l_2}^2\right)}\boldsymbol{e}_j\boldsymbol{p}_j \tag{7.32}$$

Eq. (7.25) tells that in order to compute the Hessian of the logarithmic barrier function $\hat{I}_-(g_j(\cdot);t)$ we only need its gradient given by eq. (7.32) and

$$\nabla^2(||\boldsymbol{p}_j||_{l_2}^2 - 1) = \nabla^2(||\boldsymbol{e}_j^T\boldsymbol{p}||_{l_2}^2 - 1) = 2\boldsymbol{e}_j\boldsymbol{e}_j^T \tag{7.33}$$

Having eq. (7.25), eq. (7.32) and eq. (7.33) we can show the Hessian-vector multiplication of the logarithmic barrier function $\hat{I}_-(g_j(\boldsymbol{p});t)$

$$\nabla^2 \hat{I}_-(g_j(\boldsymbol{p});t)\boldsymbol{v} = \frac{1}{t}\hat{\boldsymbol{p}}_{(j)}^2\boldsymbol{e}_j\boldsymbol{p}_j\boldsymbol{p}_j^T\boldsymbol{e}_j^T\boldsymbol{v} + \frac{1}{t}\hat{\boldsymbol{p}}_{(j)}\boldsymbol{e}_j\boldsymbol{e}_j^T\boldsymbol{v} \tag{7.34}$$

$$= \frac{1}{t}\hat{\boldsymbol{p}}_{(j)}^2\boldsymbol{e}_j\boldsymbol{p}_j\boldsymbol{p}_j^T\boldsymbol{v}_j + \frac{1}{t}\hat{\boldsymbol{p}}_{(j)}\boldsymbol{e}_j\boldsymbol{v}_j \tag{7.35}$$

where $\hat{\boldsymbol{p}}_{(j)} := \frac{2}{1-||\boldsymbol{p}_j||_{l_2}^2}$.

Let $\nu$ be the duality gap. Similarly to Kim et al. [2007], we use the following formula for the update-step

$$\text{Update}(t,\nu,\alpha) := \begin{cases} \max\left\{\mu \min\left\{\frac{2n}{\nu}, t\right\}, t\right\} & \text{if } \alpha \geq \alpha_m \\ t & \text{otherwise} \end{cases} \tag{7.36}$$

where $\alpha_m \in (0,1]$, $\mu > 1$ and $n$ is the length of vector $\boldsymbol{p}$. Moreover, the number of iterations for direction-search is limited by the adaptive rule

$$\varepsilon_{iteration} := \min\left\{0.1, \zeta\frac{\nu}{||\boldsymbol{g}||_{l_2}}\right\} \tag{7.37}$$

where $\zeta > 0$ is some parameter and $\boldsymbol{g}$ is the gradient at the current point [Kim et al. 2007]. Intuitively, we search for the direction with low accuracy at early iterations when the duality gap is quite big and gradually we increase accuracy at later iterations when the duality gap becomes smaller.

In the context of problem (7.14) we have $\boldsymbol{Q} := \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$ and $\boldsymbol{r}^T := -\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{B}^T$. Therefore Algorithm 20 can be used in order to solve problem (7.14).

[3] The standard vectors were described in Appendix E.

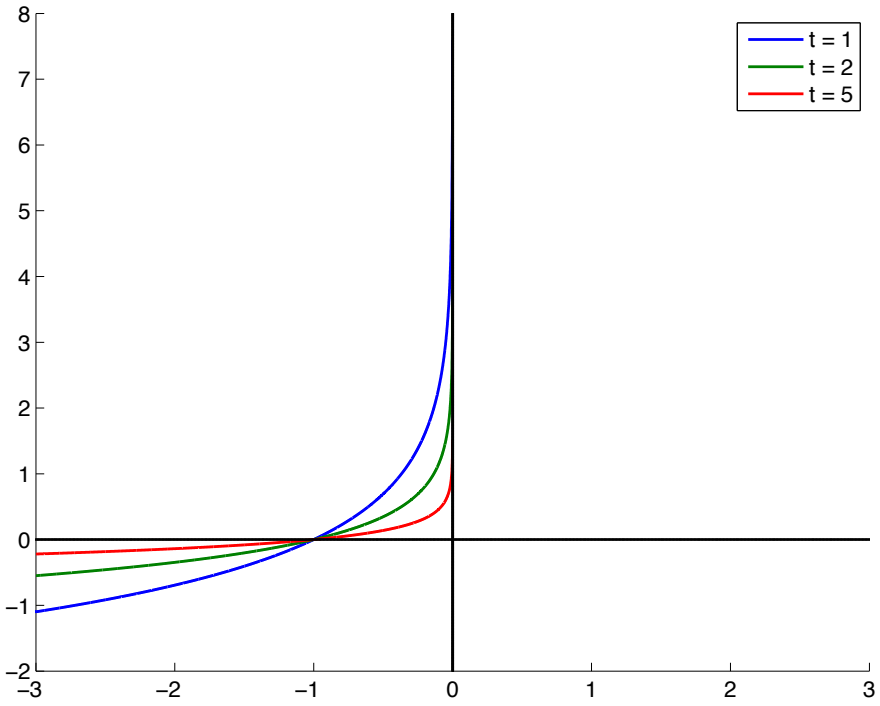



Figure 7.1: Plots of the logarithmic barrier function given by eq. (7.22) with respect to different $t$.

Chapter 8

# Experiments

## Contents



In this chapter, the setup for the experiments is presented. In the trials design matrices of two kinds have been used, the first one created by Variable density phase encoding [Lustig et al. 2007] and the second one produced by Bayesian experimental design [Seeger et al. 2009b]. Next, outcomes of the experiments are shown and a short discussion about the results of the experiments is provided.


Many thanks to Max-Planck-Institut für Biologische Kybernetik in Tübingen[1] for providing data for our experiments.


Section 8.1 contains the overview of the experiments. It also briefly recalls algorithms, explains how we measured the complexity and how the graphs were produced. Section 8.2 presents the setup of the experiments with the MRI energy function used as the objective function. In particular free parameters that were found to work well with the problem are provided. Section 8.3 presents discussion and outcomes providing that the MRI energy function was used as the objective function. Section 8.4 presents the setup of the experiments with the MRI elastic energy function used as the objective function. Again, free parameters that were found to work well with the problem are provided. Section 8.5 presents discussion and outcomes providing that the MRI elastic energy function was used as the objective function.

## 8.1 Introduction

We took images of the vertical profiles of the four subjects (so called sagittal MR images). Each image contains 256 rows and 256 columns. Next, we chose the MR images for each

[1] http://www.kyb.mpg.de/.

subject that represent two neighbouring slices (number 8 and number 9) [2]. Every MR image was corrupted by the phase noise during the acquisition process and next subsampled in the Fourier-space [Seeger et al. 2009b].

Mathematically this setting can be modeled as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{u}_{true}$$

where $\boldsymbol{u}_{true}$ is the original MR image (already containing noise) and $\boldsymbol{y}$ is the subsampled image (our observation). The subsampling Fourier operator (Definition 3.3) was used as the measurement matrix $\boldsymbol{X}$. Two different subsampling schemes were chosen, one produced by the Bayesian experimental design proposed in Seeger et al. [2009b], and the second produced by the Variable density phase encoding proposed in Lustig et al. [2007]. In order to obtain the image as close to the original as possible, two reconstruction models were established: model (3.7) and model (7.2).

Three algorithms that deal with model (3.7), namely *FISTA*, *Augmented Lagrangian* and *Non-linear Conjugate Gradient* were compared. Although, for the sake of brevity we call first two methods *FISTA* and *Augmented Lagrangian*, we actually refer to the algorithms that were described in Subsection 6.2.9.2 and Subsection 6.3.4 (Figure 8.3 shows the difference between two versions of *Augmented Lagrangian*, in the rest of the experiments Algorithm 15 was used). All three algorithms differ slightly from each other. For instance, *Non-linear Conjugate Gradient* uses the differentiable surrogate $||\boldsymbol{u}||_\varepsilon := \sum_j \sqrt{||\boldsymbol{u}_j||_{l_2}^2 + \varepsilon}$ of the $l_1$-norm, *Augmented Lagrangian* splits the variables in the objective function in order to solve easier subproblems, whereas *FISTA* exploits the duality and the proximity operator to tackle the problem. Despite differences all three methods perform only two fast Fourier transforms per iteration.

We also compared two algorithms that deal with model (7.2). Both algorithms were presented in Section 7.3 and Section 7.4. For the sake of brevity, by *Quadratic FGP* and *Interior Point* we refer to the *Normed Constrained Quadratic FGP* method and the *Truncated Newton Primal Dual Interior Point* method respectively. *Quadratic FGP* is a first order method resembling *FISTA*, whereas *Interior Point* is a second order method that exploits the information about the Hessian in order to solve the problem. Every iteration of the *Interior Point* has to solve the linear system (7.27) in order to obtain a search direction. We used Matlab function *pcg*[3] to solve this system.

To make the comparison, we have to also specify how we measured the complexity of the algorithms. Note that $c(\boldsymbol{X}\boldsymbol{u}) = O(n\log(n))$ and $c(\boldsymbol{B}_a\boldsymbol{u}) = c(\boldsymbol{B}_r\boldsymbol{u}) = c(\boldsymbol{B}_i\boldsymbol{u}) = O(n)$ (the implementation of the operators was briefly described in Subsection 3.2.7), where $n$ is the length of the vector $\boldsymbol{u}$ and $c$ is the time complexity with respect to $n$. Since all algorithms perform only matrix-vector multiplication operations we can say that the cost of one iteration is dominated by computing $\boldsymbol{X}\boldsymbol{u}$, which, in turn, amounts to executing one fast Fourier transform (FFT). The cost of dropping columns of the transformed image is negligible in comparison to computing FFT of the image. Thus, in this thesis, we measured

[2] The reader who is not familiar with the details of the setup can assume that two neighbouring and middle slices were taken.

[3] Preconditioned conjugate gradients method http://www.mathworks.com/help/techdoc/ref/pcg.html.

the complexity of the algorithms by the number of performed FFT. Moreover, fft is an unit that corresponds to performing exactly one FFT.

In the y-axis of the graph we measured $l_2$ distance between $\boldsymbol{u}^\star$ and $\boldsymbol{u}_{true}$. More precisely, we measured the error defined as

$$error\ against\ true\ solution = |||\boldsymbol{u}^\star| - |\boldsymbol{u}_{true}|||_{l_2} \tag{8.1}$$

where $\boldsymbol{u}^\star$ is the reconstruction obtained by the algorithm and $\boldsymbol{u}_{true}$ is the image used to generate the subsampled image $\boldsymbol{y}$. In the x-axis we measured the complexity of the algorithms. All results were averaged over two slices and over four subjects in order to eliminate bias that potentially may arise between some methods, models and observations.

In order to visualize the complex-valued images, we computed absolute value of their coefficients.

## 8.2 Setup - MRI Energy Function

### 8.2.1 Bayesian Experimental Design

We conducted experiments using the Bayesian experimental design [Seeger et al. 2009b] together with four different design sizes[4]: 48, 64, 96, 128. Hyperparameters were set to be $\tau_a := 0.07$, $\tau_r := 0.04$, $\tau_i := 0.1$, $\sigma^2 := 2.4 \cdot 10^{-4}$ and $\forall_{l \in \{a,r,i\}}\ \kappa_l := \sigma\tau_l$ [Seeger and Nickish 2011]. The Daubechies 4 wavelets were chosen as the wavelet-part regularization term.

We compared the following methods: *FISTA*, *Augmented Lagrangian* and *Non-linear Conjugate Gradient*. All algorithms start from zero vectors, that is $\boldsymbol{u}^{start} := \boldsymbol{0}$, $\boldsymbol{s}_{(j)}^{start} := \boldsymbol{0}$, $\boldsymbol{r}^{start} := \boldsymbol{0}$, $\boldsymbol{d}_{(j)}^{start} := \boldsymbol{0}$ for all $j \in \{a, r, i\}$.

The $\varepsilon$ parameter in $||\cdot||_\varepsilon$ for *Non-linear Conjugate Gradient* was fixed to be $\varepsilon := 4 \cdot 10^{-12}$. In the case of *Augmented Lagrangian*, we found that extra parameters set to $\mu_a := 20k_a$, $\mu_r := 20k_r$ and $\mu_i := 20k_i$ where $k_l := \sigma\tau_l$ for $l \in \{a, r, i\}$ work fine. *FISTA* requires in addition the number of inner iterations to be set. Since we want to keep computing $\boldsymbol{Xu}$ as a dominating factor we cannot allow the number of inner iterations to be too large. For the purpose of these experiments *FISTA* performs exactly 3 inner iterations.

### 8.2.2 Variable Density Phase Encoding

Now we consider the Variable density phase encoding proposed in Lustig et al. [2007]. The setting is the same as in the case of the Bayesian experimental design, however, this time in addition we also averaged over ten randomly generated matrices. The results are shown in Figures H.13, H.14, H.15 and H.16. Error bars indicate deviations from the average. Figures H.17, 8.7, H.19, H.20 show the average plots over ten randomly generated matrices, two slices and four subjects. Algorithm 21 presents this process of averaging in a formal way. The following list shows the meaning of functions and constants used in this algorithm:

- *take_observation*$(j,\ k,\ l)$ takes the observation of the subject number $k$, the slice number $l$, and where the $j$-th measurement matrix was used.

[4] In our setting the design size is the cardinality of the set of chosen indices divided by the number of rows. In the experiments the design size is $N_{design} := \frac{|\mathcal{J}|}{256}$ (see also Definition 3.1 in Subsection 3.2.2).

- $perform\_reconstruction(observation,\ j)$ returns an array of reconstructions. Element $d$ in this array is a reconstruction obtained after $d$ iterations.
- $compute\_error(trueImage,\ reconstructions)$ returns an array of errors computed between the true image and every reconstruction from the array $reconstructions$. The errors were computed by using eq. (8.1).
- $\max(errorBarsUp,\ cumulativeError)$ returns the piecewise maximum between vectors: $errorBarsUp$ and $cumulativeError$.
- $\min(errorBarsDown,\ cumulativeError)$ returns the piecewise minimum between vectors: $errorBarsDown$ and $cumulativeError$.
- $N_m$ is the number of measurement matrices.
- $N_s$ is the number of subjects.
- $N_l$ is the number of slices.

**Algorithm 21** Averaging of Results

$averageError := [0, 0, \ldots, 0]^T$
$errorBarsUp := -[\inf, \inf, \ldots, \inf]^T$
$errorBarsDown := [\inf, \inf, \ldots, \inf]^T$
**for** $j = 0, 1, \ldots, N_m$ **do**
  $cumulativeError := [0, 0, \ldots, 0]^T$
  **for** $k = 0, 1, \ldots, N_s$ **do**
    **for** $l = 0, 1, \ldots, N_l$ **do**
      $observation = take\_observation(j,\ k,\ l)$
      $reconstructions = perform\_reconstruction(observation,\ j)$
      $err = compute\_error(trueImage,\ reconstructions)$
      $cumulativeError := cumulativeError + err$
    **end for**
  **end for**
  $averageError := averageError + cumulativeError$
  $errorBarsUp := \max(errorBarsUp,\ cumulativeError)$
  $errorBarsDown := \min(errorBarsDown,\ cumulativeError)$
**end for**
$averageError := \frac{averageError}{N_m \cdot N_s \cdot N_l}$
$errorBarsUp := \frac{errorBarsUp}{N_s \cdot N_l}$
$errorBarsDown := \frac{errorBarsDown}{N_s \cdot N_l}$

Since every trial may result in a different design matrix, we have to specify how the image is shown. Let the median image denote the reconstructed image which corresponds to the median error[5]. If the median error consists of two values, the median image is the

[5] Here, if $e_i \leq e_{i+1}$ then $\text{median}(\{e_1, e_2, \ldots, e_{2k-1}\}) = \{e_k\}$ and $\text{median}(\{e_1, e_2, \ldots, e_{2k}\}) = \{e_k, e_{k+1}\}$.

one which additionally yields the reconstruction error that is closer to the average. This uniquely defines the median image. Figures 8.8, 8.9, 8.10 and 8.11 show median images for different design sizes.

Let the difference image be an image obtained from the absolute difference between the true image and the reconstruction. Figures H.22, H.24, H.26 and H.28 show the difference images where either the Bayesian experimental design or the Variable density phase encoding was used.

While *Augmented Lagrangian* produced good results after 10 iterations if the Bayesian experimental design was used, for the Variable density phase encoding this number had to be increased to 30. Then we could compare the reconstructions given by the Bayesian experimental design with the reconstructions given by the Variable density phase encoding.

## 8.3 Results and Discussion - MRI Energy Function

Figure 8.1 shows that *Augmented Lagrangian* converges faster than the other methods. In addition, it gives very good results after only 20 ffts, better than results of the other methods (see Figure 8.4). The *Non-linear Conjugate Gradient* method also performs quite well; after exactly 20 ffts it gives result around 1.0 unit of the $l_2$-error better than *FISTA*. However, if we allow the solvers to work for longer than 60 FFTs then their performance becomes almost the same and the advantage of using one of them is not so clear. Other experiments (Figures H.1, H.3, H.4) also confirm this claim.

Figure 8.5 shows the impact of the design size on the reconstruction. Clearly, the bigger design size was chosen the more details were added and the overall reconstruction was improved which results in smaller errors (see also Figures H.1, 8.1, H.3, H.4). So, in many applications, we can easily go to 1/2 of the Nyquist rate which corresponds to the design size $N_{design} = 128$ (the top-right image in Figure 8.5) and sometimes even go beyond that to 1/4 of the Nyquist rate ($N_{design} = 64$, second-left image in Figure 8.5).

All parameters in both methods *FISTA* and *Non-linear Conjugate Gradient* have some interpretation: *Non-linear Conjugate Gradient* depends on the precision $\varepsilon$ and *FISTA* depends on the step-size $\alpha$ which can be also interpreted as a reciprocal of the upper bound of the Lipschitz constant. However, the meaning of extra parameters in *Augmented Lagrangian* method is not so clear. Therefore special care must be taken in order to fix them. Figure 8.2 shows that *Augmented Lagrangian* is sensitive to changing its parameters. We chose the following parameters:

- Setting 1: $\mu_a := 100 \cdot \kappa_a$, $\mu_r := 100 \cdot \kappa_r$, $\mu_i := 3 \cdot \kappa_i$.
- Setting 2: $\mu_a := 610 \cdot \kappa_a$, $\mu_r := 420 \cdot \kappa_r$, $\mu_i := 70 \cdot \kappa_i$.
- Setting 3: $\mu_a := 20 \cdot \kappa_a$, $\mu_r := 20 \cdot \kappa_r$, $\mu_i := 20 \cdot \kappa_i$.

to illustrate the impact of these parameters on the algorithm.

Figure 8.6 show the naive reconstruction which amounts to filling the missing columns with zeros in the Fourier space. Clearly, the naive reconstruction doesn't give good results. See also additional Figures H.9, H.10, H.11, H.12.

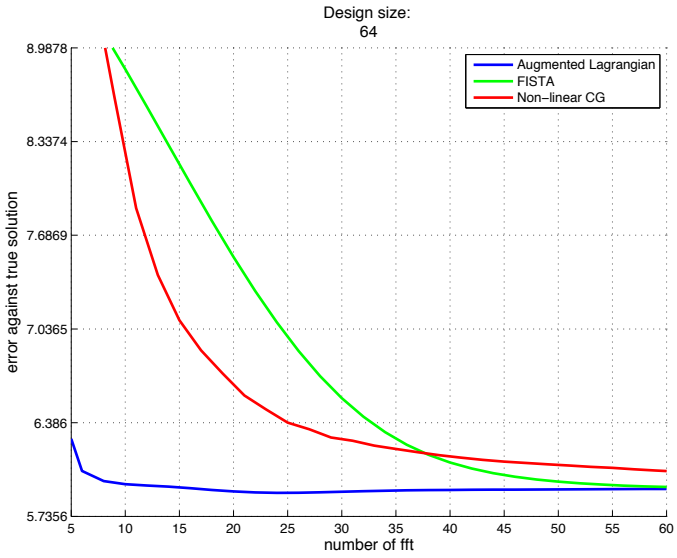



Figure 8.1: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

Figure 8.7 shows that the performance of *FISTA* and *Non-linear Conjugate Gradient* per fixed design size is also very similar in the case of using the Variable density phase encoding. Also in this case, the *Augmented Lagrangian* method outperforms other methods (see also Figures H.17, H.19, H.20, and Figures H.13, H.14, H.15, H.16). It suggests that the observations where the Bayesian experimental design was used are also valid for a different measurement matrix which was given by Lustig et al. [2007].

Moreover, the Variable density phase encoding doesn't perform well when small design sizes were used and the advantage of using the Bayesian experimental design over the Variable density phase encoding is noticeable in these cases (Figures 8.8 and 8.9). Using bigger design sizes makes this advantage less visible (Figures 8.10 and 8.11).

Figures H.22, H.24, H.26 and H.28 show that the difference between the true image and the reconstruction is more structured in the case of the smaller design sizes.

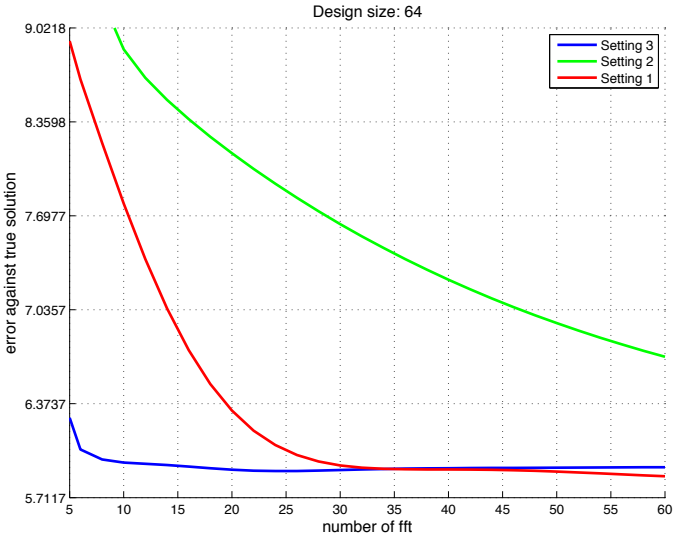



Figure 8.2: Design size: 64. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ produced by *Augmented Lagrangian* with different parameters (averaged over two slices and four different subjects).

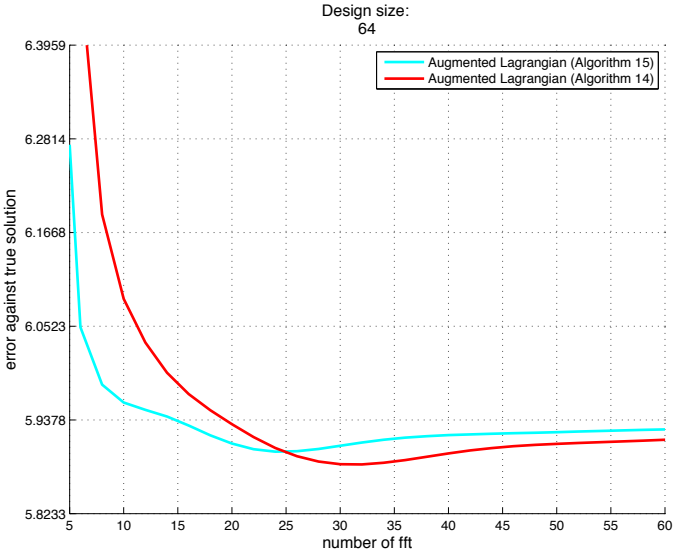



Figure 8.3: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).


Figure 8.4: Design size: 64. Shown are outcomes produced by all three methods. The number of iterations was limited to 10 steps. The Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.

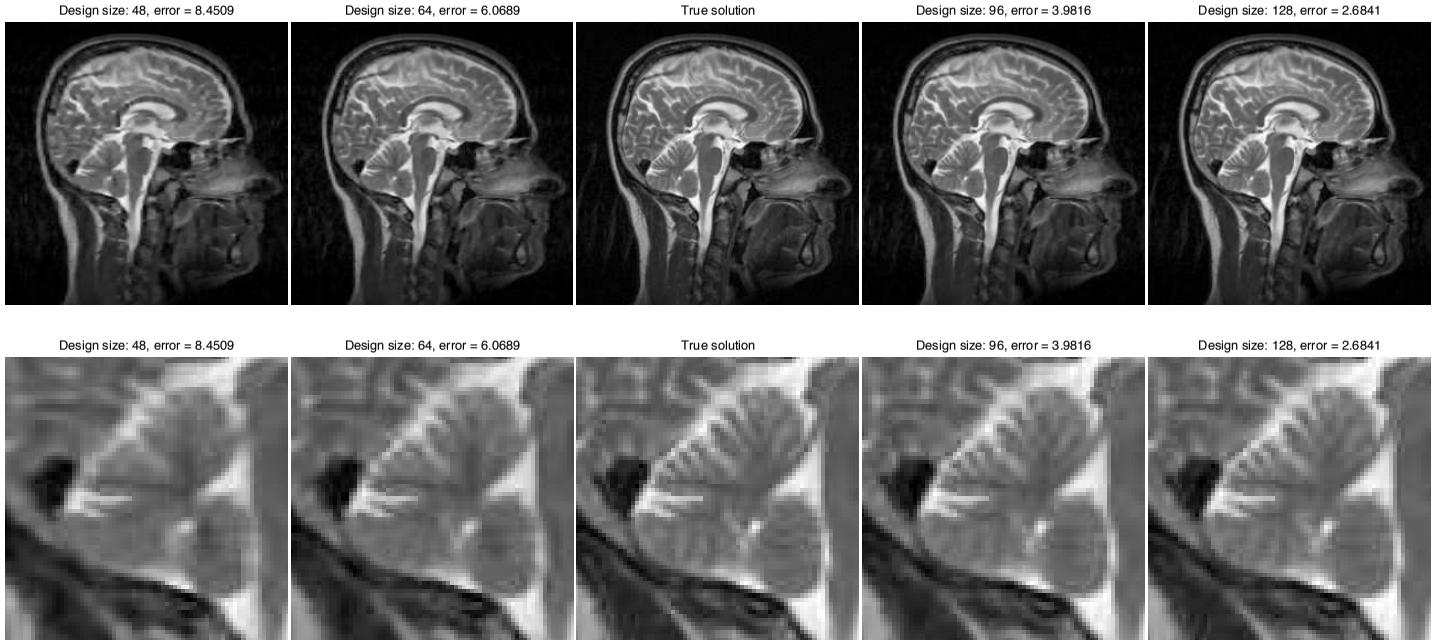
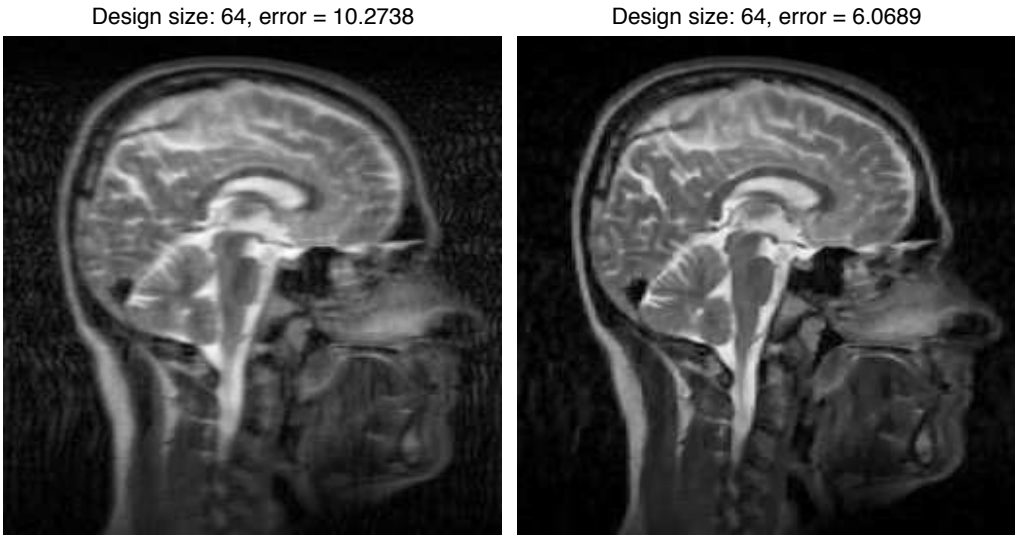



Figure 8.5: Results for the Bayesian experimental design with different design sizes 48, 64, 96, 128 (TE=92ms, subject 2, slice 8, sagittal image) together with true image $\boldsymbol{u}_{true}$. The number of iterations was limited to 10 steps. From the left $N_{design} = 48$, $N_{design} = 64$, true image, $N_{design} = 96$, $N_{design} = 128$. Reconstructions were obtained by using *Augmented Lagrangian*. Upper row: Full images. Lower row: Blow-ups.




(a) Naive reconstruction (b) Reconstruction with model (3.7)

Figure 8.6: Design size: 64. The reconstruction with model (3.7) was done by *Augmented Lagrangian*. The number of iterations was limited to 10 steps. Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.

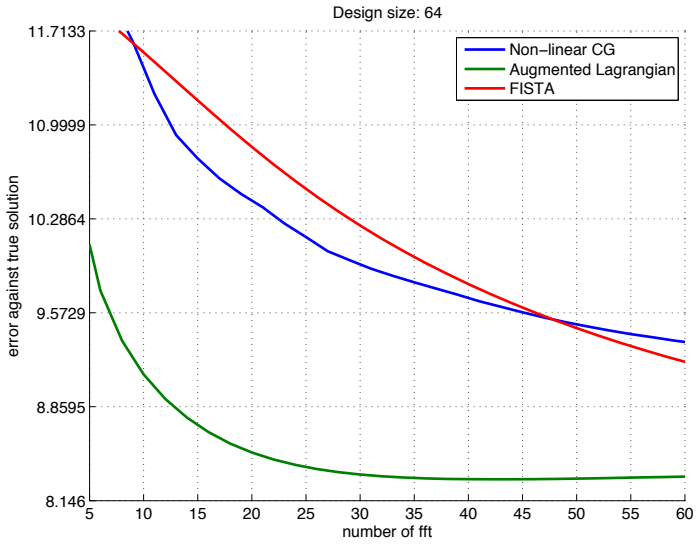
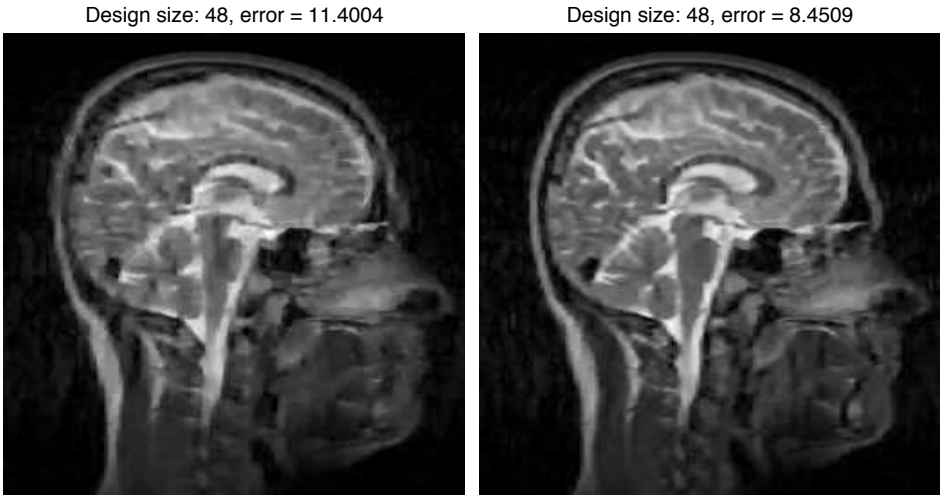
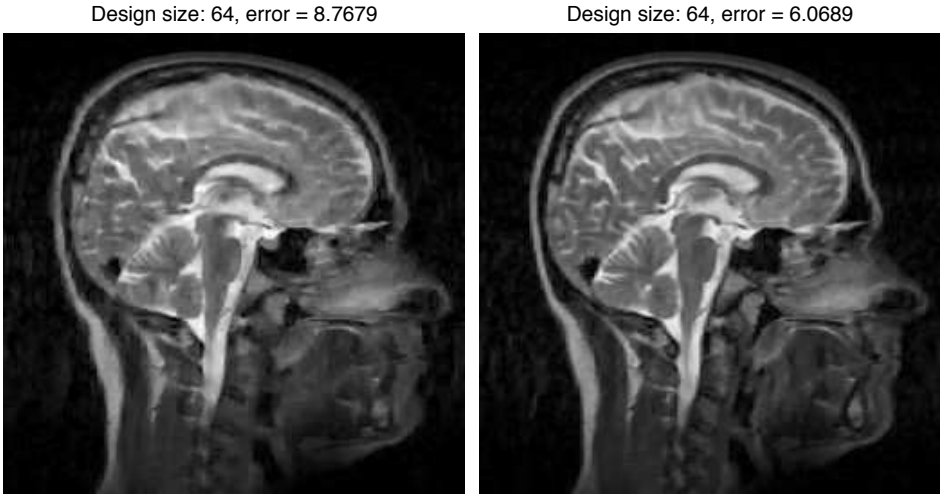



Figure 8.7: Design size: 64. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices).



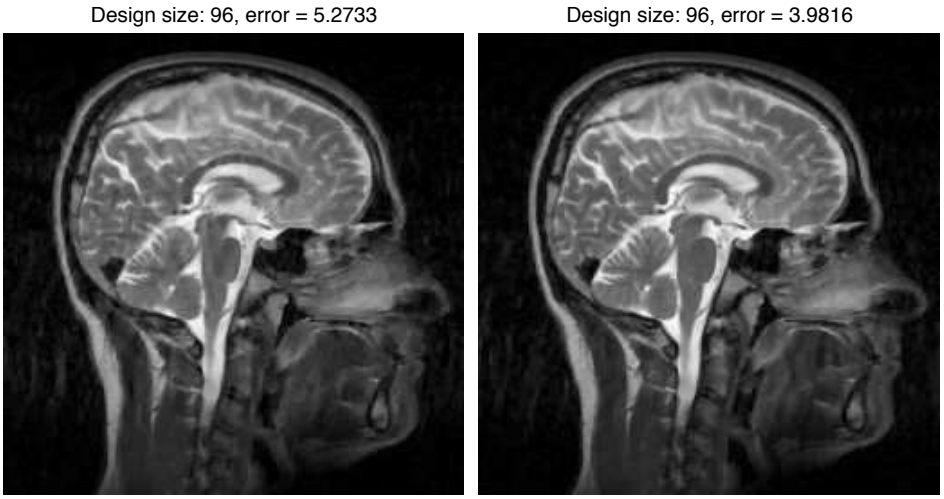

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure 8.8: Design size: 48. The median image for the Variable density phase encoding is shown in Figure 8.8a and the reconstructed image for the Bayesian experimental design is shown in Figure 8.8b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.



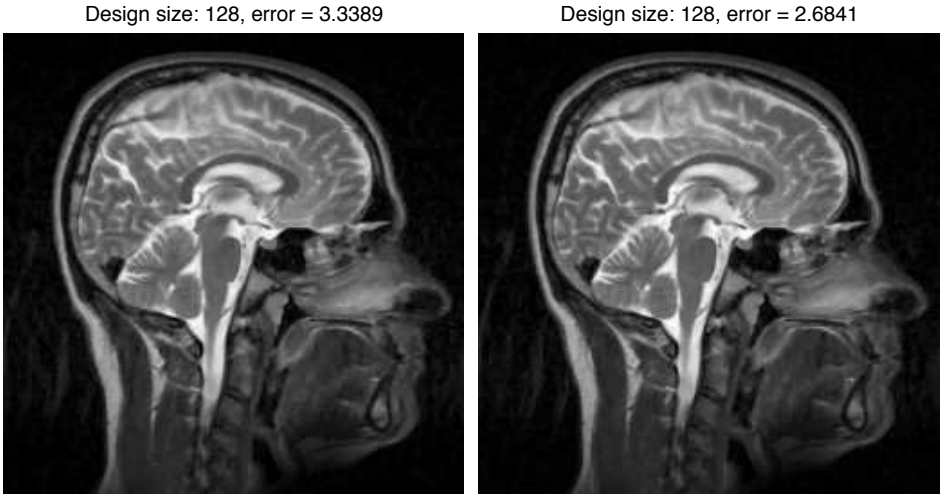

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure 8.9: Design size: 64. The median image for the Variable density phase encoding is shown in Figure 8.9a and the reconstructed image for the Bayesian experimental design is shown in Figure 8.9b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure 8.10: Design size: 96. The median image for the Variable density phase encoding is shown in Figure 8.10a and the reconstructed image for the Bayesian experimental design is shown in Figure 8.10b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure 8.11: Design size: 128. The median image for the Variable density phase encoding is shown in Figure 8.11a and the reconstructed image for the Bayesian experimental design is shown in Figure 8.11b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.

## 8.4 Setup - MRI Elastic Energy Function

### 8.4.1 Bayesian Experimental Design

We conducted experiments using the Bayesian experimental design [Seeger et al. 2009b] together with four different design sizes: 48, 64, 96, 128. Hyperparameters were set to be $\tau_a := 0.07$, $\tau_r := 0.04$, $\tau_i := 0.1$, $\sigma^2 := 2.4 \cdot 10^{-4}$ and $\forall_{l \in \{a,r,i\}}\ \kappa_l := \sigma\tau_l$ [Seeger and Nickish 2011]. The Daubechies 4 wavelets were chosen as the wavelet-part regularization term. The elastic parameters were set to be $\mu_a := 10^{-3}$, $\mu_r := 10^{-3}$ and $\mu_i := 10^{-3}$.

Two algorithms were employed in the experiments: *Quadratic FGP* and *Interior Point*. Both algorithms were described in Chapter 7. All algorithms start from zero vectors, that is $\boldsymbol{p}^{start} := \mathbf{0}$. Since the found upper bound of the Lipschitz constant was too big to be useful in practice, we chose experimentally the step-size $\alpha$ in *Quadratic FGP* to $\alpha := 700$. The backtracking (Algorithm 22) was embedded in *Interior Point* in order to perform the line-search. Parameters for the line-search were chosen as follows: $\alpha_0 := 1.0$, $\beta := 0.5$. We set the parameters in the update function (7.36) as it was done in Kim et al. [2007]: $\mu := 2.0$, $\alpha_m := 0.5$. Parameter $\zeta$ in the adaptive rule (7.37) was set to be 0.01. We used Matlab function *pcg*[6] to perform *Conjugate Gradient* iterations. The truncation rule in our implementation of the *Interior Point* method is as follows: we stop cg-iterations whenever the cumulative number of the cg-iterations exceeds the maximal allowed number of cg-iterations, or we computed a point with a relative tolerance less than $\varepsilon_{iteration}$ (7.37). We also set $t_0$, needed by the update rule 7.36, by using similar rule to Kim et al. [2007], that is

$$t_0 := \frac{1}{\tau_a \cdot \sigma}$$

In the main experiments, we set the maximal number of cg-iterations to be 600. The rationale behind this number is simple:

- On the one hand we expect from the adaptive rule (7.37) to decide when to stop cg-iterations.
- On the other hand we stop cg-iterations if there are too many of them.

Although, one could limit the cg-iterations even more to make this method faster, we found that in this case the method produces the results with lower quality (Figures 8.13 and 8.14). This happens since *Interior Point* converges prematurely if the number of cg-iterations is not big enough.

## 8.5 Results and Discussion - MRI Elastic Energy Function

Figure 8.12 shows the average performance of two algorithms *Quadratic FGP* and *Interior Point* providing the design size was chosen to be 64. The former algorithm outperforms the latter. The reason for the bad performance of *Interior Point* was the number of steps

[6] http://www.mathworks.com/help/techdoc/ref/pcg.html.

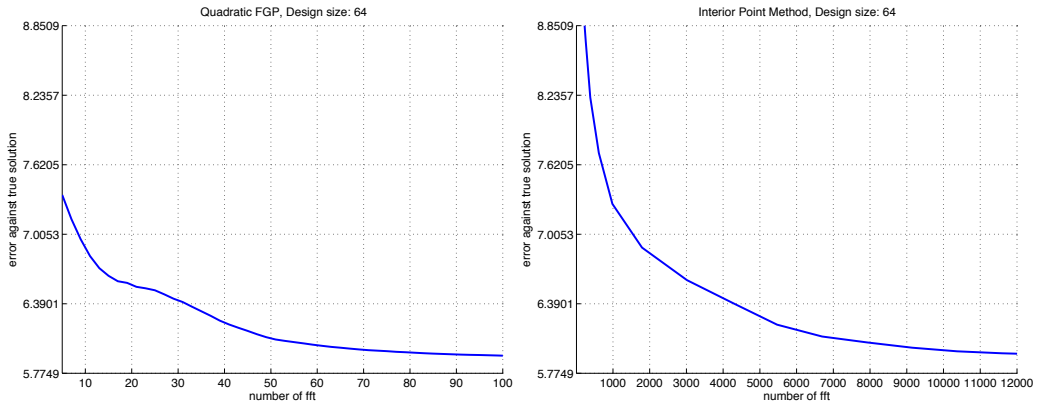



Figure 8.12: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects). The elastic model (7.2) was used.

that *Conjugate Gradient* had to execute in order to find a good direction. The first order method *Quadratic FGP* also outperforms *Interior Point* when other design sizes were chosen (Figures H.29, H.31, H.32).

Figures 8.13 and 8.14 show that if the cg-iterations in the *Interior Point* method are truncated too early, then the algorithm may converge prematurely which results in the reconstruction of lower quality. On the other hand, the more cg-iterations, the larger number of performed FFT, and so the worse complexity of the algorithm.

Figure 8.15 shows the performance of *Augmented Lagrangian* which works with model (3.7) and the performance of *Quadratic FGP* which works with model (7.2). We can see that *Augmented Lagrangian* converges faster than *Quadratic FGP* (see also Figures H.33, H.35 and H.36).

Figure 8.16 shows the reconstruction for two models, the original model (3.7) and its elastic extension given by model (7.2). *Augmented Lagrangian* was employed to find the reconstruction where the former model was used and *Quadratic FGP* was used to find the reconstruction where the latter model was chosen. The reconstruction where the original model was used yields slightly smaller errors. In addition, the original model is simpler and presented methods which work with the original model are faster. Therefore, the original model is slightly more preferred. (see also Figures H.37, H.39 and H.40).

Figure 8.17 shows the impact of the design size on the reconstruction. Similarly to the case when model (3.7) was used, we can see the monotonic impact of the design size. That is, the more lines that were added the smaller the error obtained (see also Figures H.29, H.31, H.32).

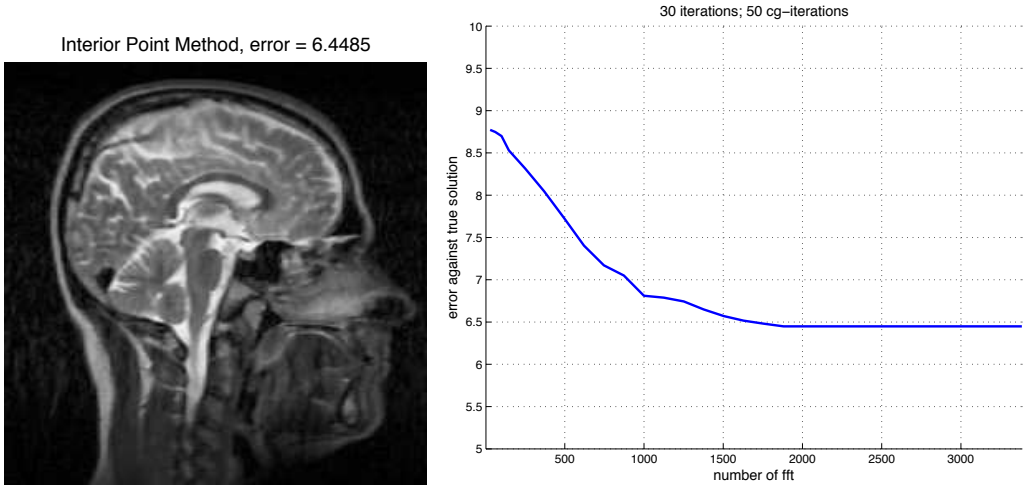



Figure 8.13: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. The elastic model (7.2) was used. The number of iterations was limited to 30 steps. The maximal number of cg-iterations was limited to 50 steps. Image description: TE=92ms, subject 2, slice 8, sagittal image.

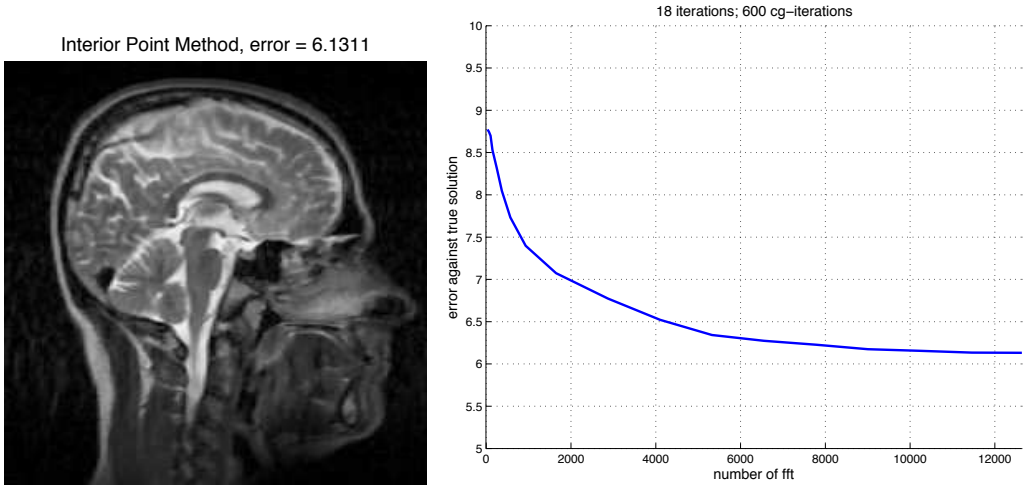



Figure 8.14: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. The elastic model (7.2) was used. The number of iterations was limited to 18 steps. The maximal number of cg-iterations was limited to 600 steps. Image description: TE=92ms, subject 2, slice 8, sagittal image.

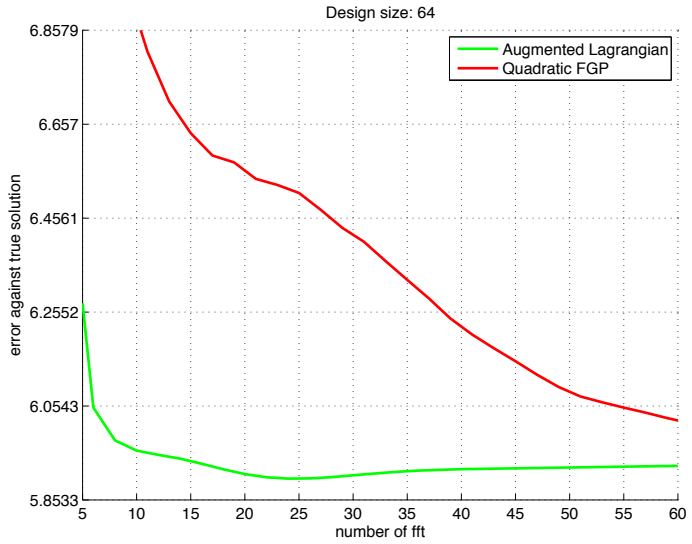
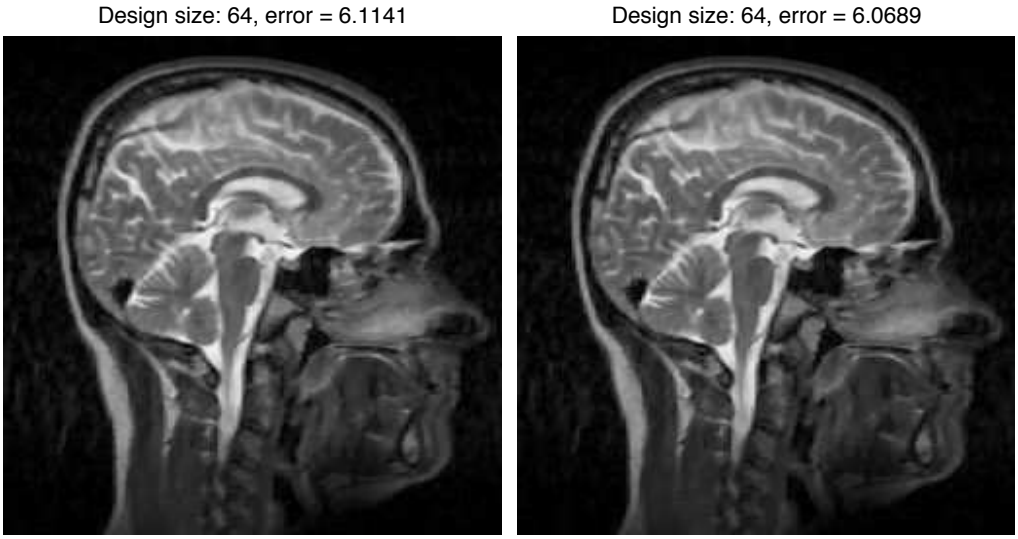



Figure 8.15: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. Augmented Lagrangian was used to solve Problem (3.7). Quadratic FGP was used to solve Problem (7.2).




(a) Elastic extension

(b) Original model

Figure 8.16: Design size: 64. The Bayesian experimental design was used. Figure 8.16a shows the reconstruction made by *Quadratic FGP* with model (7.2). Figure 8.16b shows the reconstruction made by *Augmented Lagrangian* with model (3.7). Image description: TE=92ms, subject 2, slice 8, sagittal image.

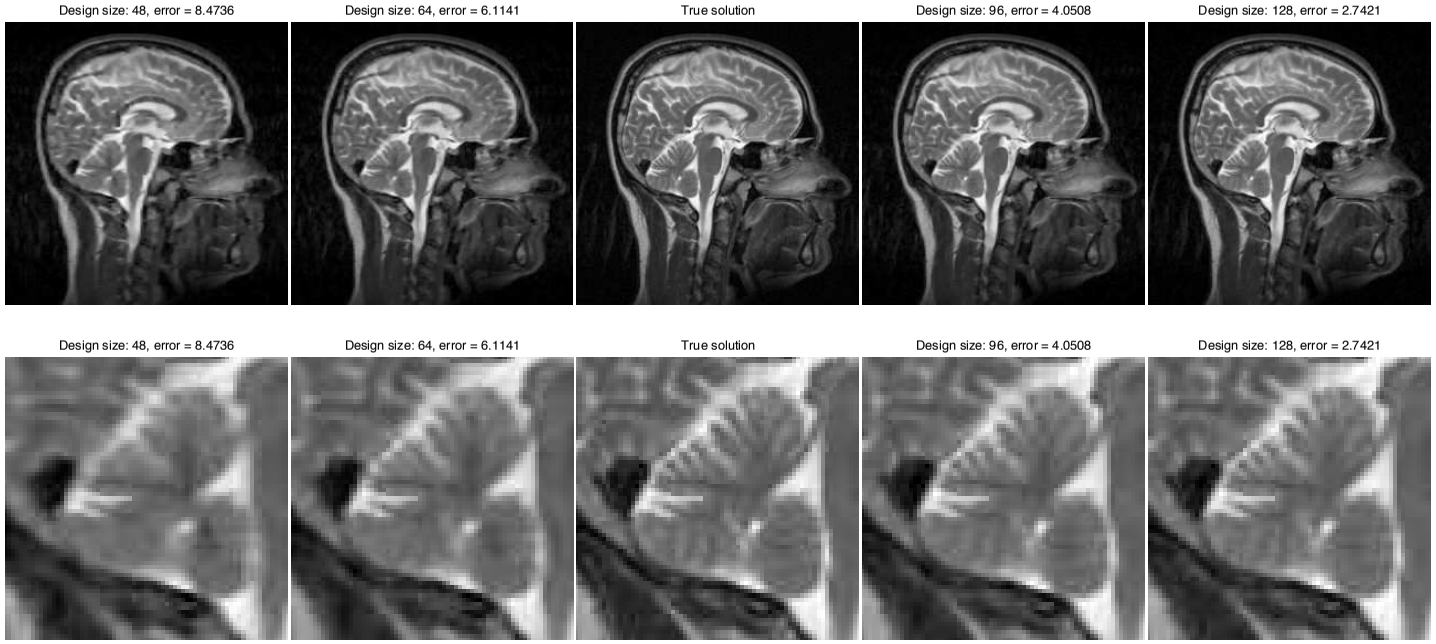



Figure 8.17: Results for the Bayesian experimental design with different sizes 48, 64, 96, 128 (TE=92ms, subject 2, slice 8, sagittal image) together with true image $\boldsymbol{u}_{true}$. Number of iterations was limited to 60 steps. From the left $N_{design} = 48$, $N_{design} = 64$, true image, $N_{design} = 96$, $N_{design} = 128$. For the reconstruction *Quadratic FGP* method was used. Upper row: Full images. Lower row: Blow-ups.

Chapter 9

# Summary and Future Work

## Contents



## 9.1 Summary

In Chapter 2, we explained how the images were represented. We also briefly and informally presented the concept of sparsity, a sparsifying transform and compressed sensing.

In Chapter 3, we introduced the optimization problem that is the central model of this thesis. This model consists of the data fitting term and the regularization term. The latter is a linear combination of $||\boldsymbol{B}_a\boldsymbol{u}||_{l_1}, ||\boldsymbol{B}_r\boldsymbol{u}||_{l_1}, ||\boldsymbol{B}_i\boldsymbol{u}||_{l_1}$ where $\boldsymbol{B}_a$ is an orthonormal wavelet transform, $\boldsymbol{B}_r$ is a finite difference operator and $\boldsymbol{B}_i$ is the imaginary part penalizing operator. These operators were also described in that chapter.

In Chapter 4, we defined the elastic linear system. A special case of this system occurs frequently in this thesis and can be solved efficiently in the Fourier domain.

In Chapter 5, we defined the projection and its generalization called the proximity operator. We also showed a few examples of the proximity operator that play important role in this thesis. Moreover, we can compute the solution of the denoising problem which is an instance of the proximity operator very efficiently by utilizing the soft-thresholding operator.

In Chapter 6, we derived and presented two algorithms that are based on *FISTA* and *Augmented Lagrangian* frameworks. Both algorithms belong to the class of the first order methods and exploit only the information about the gradient to solve the optimization problem. On the one hand both methods share the same complexity - two invocations of the fast Fourier transform per iteration. On the another hand, they use a different approach to tackle the problem. In addition, another first order method, called *Non-linear Conjugate Gradient*, was briefly described in that chapter.

In Chapter 7, we introduced the elastic extension of the original problem by adding $l_2$-norm regularization. This leads to the invertible elastic matrix. Next, we showed dualization of the problem which turns out to be an instance of the normed constrained quadratic programming. The dual problem is also a working space for two algorithms, the first order method called *Normed Constrained Quadratic Fast Gradient Projection* and the second order method called *Truncated Newton Primal Dual Interior Point*.

In Chapter 8, all algorithms presented in this thesis against the problem of the reconstruction of MR images from incomplete measurements were empirically confronted.

In the experiments conducted and presented in Section 8.3, it was shown that two methods derived in Chapter 6, the *Non-linear Conjugate Gradient* method and the method based on the *FISTA* framework, performed almost equally well given our settings. However, the better performance of the third presented method that was based on *Augmented Lagrangian* is visible. In that section the impact of the design size on the reconstruction was also shown. Basically, the more lines in the Fourier space dropped, the bigger the error becomes. In addition, we also presented the importance of the model. If the naive reconstruction which amounts to filling the missing lines with zeros in the Fourier space was used, the reconstruction became much worse. Finally, we showed the impact of different measurement matrices on the reconstruction error. In the experiments the measurement matrices of two kinds were used: the first one produced by the Bayesian experimental design and the second one produced by the Variable density phase encoding.

In the experiments conducted and presented in Section 8.5, it was shown that the first order method called *Normed Constrained Quadratic FGP* surpassed the second order method called *Truncated Newton Primal Dual Interior Point* in terms of the performance. However, the bad performance of the latter method was mainly caused by the large number of iterations required to find a good direction. This suggests that preconditioning may be necessary to obtain better performance of *Truncated Newton Primal Dual Interior Point* and it is itself an interesting future extension to the algorithm.

Finally, we compared the reconstructions based on the original model with the reconstructions based on the elastic extension to the original model. Although, both reconstructions were very similar to each other, and produced similar errors, since the original model is simpler it is slightly more preferred.

## 9.2 Future Work

There are a few interesting directions that can extend the work done in this thesis. We showed worse performance of *Truncated Newton Primal Dual Interior Point* than *Normed Constrained Quadratic FGP*. However, this bad performance of the former can be explained by the large number of iterations that is required to solve the linear system (7.19). Therefore, by considering different preconditioning concepts we may speed up the whole method.

Next, if we adapt *Non-linear Conjugate Gradient* to work with model (7.2) then we could compare this method to *Normed Constrained Quadratic FGP* and *Truncated Newton Primal Dual Interior Point*. The results of this comparison would be an interesting extension of this thesis.

In the experiments we used *Normed Constrained Quadratic FGP* with some fixed step-size to optimize problem (7.2). Although this step-size works well in our setting its inverse might not be an upper bound of the Lipschitz constant. Therefore it would be interesting to find an upper bound which is also useful in practice.

Finally, in the whole thesis the equispaced Fourier transform was used. However, the measurements may not be taken on the grid. For instance, one may consider spiral sampling

[Yudilevich and Stark 1988]. Therefore it would be interesting to compare the methods described in this thesis against the same optimization problems but with different sampling schemes.

# Bibliography


Manya V. Alfonso, José M. Bioucas-Dias, and Mário A. T. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *Submitted to the IEEE Transactions on Image Processing*, 2009. 4, 5, 58, 60

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009a. 4, 5, 46, 48, 51, 57

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18, 2009b. 4, 5, 47, 50, 51, 54

Matt A. Bernstein, Kevin F. King, and Xiaohong Joe Zhou. *Handbook of MRI Pulse Sequences*. Elsevier Academic Press, 1st edition, 2004. 4, 25

Dimitri P. Bertsekas. *Convex Optimization Theory*. Athena Scientific, 2009.

Dimitri P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 1999.

José M. Bioucas-Dias and Mário A. T. Figueiredo. An iterative algorithm for linear inverse problems with compound regularizers. *IEEE International Conference on Image Processing - ICIP'2008*, 2008. 4, 58

José M. Bioucas-Dias and Mário A. T. Figueiredo. A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 2007. 4, 57

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 3, 4, 49, 58, 71, 115

Emmanuel Candés, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 2006. 2

Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20:89–97, 2004. 50

Antonin Chambolle. Total variation minimization and a class of binary mrf models. *Lectures Notes in Computer Science*, 3757:136–152, 2005.

Scott S. Chen, Dave Donoho, and Michael Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 1999. 25

Patrick L. Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, 2010. 38, 39, 60

Patrick L. Combettes and Valérie R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Journal on Multiscale and Simulation*, 4:1168–1200, 2005. 39

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, 2010.

David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 2006. 2

Ivar Ekeland and Roger Témam. *Convex Analysis and Variational Problems*. Society for Industrial and Applied Mathematics, 1999. 4

Mário A. T. Figueiredo, José M. Bioucas-Dias, and Manya V. Alfonso. Fast frame-based image deconvolution using variable splitting and constrained optimization. *IEEE Workshop on Statistical Signal Processing - SSP' 2009*, 2009. 4, 5

Tom Goldstein and Stanley Osher. The split bregman method for l1 regularized problems. *SIAM Journal on Imaging Sciences*, 2:323–343, 2009. 4

Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1:606–617, 2007. 5, 73, 74, 89

Michael Lustig, Dave Donoho, and John M. Pauly. Sparse mri: The application of compressed sensing for rapid mri imaging. *Magnetic Resonance in Medicine*, 58, 2007. 2, 5, 11, 25, 26, 44, 77, 78, 79, 82

Stephane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press; 3 edition, 2008. 20, 119

Thomas P. Minka. Old and new matrix algebra useful for statistics, 2001. 6, 123, 124

Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bull. Soc. Math. France*, 93:273–299, 1965.

Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999. 5, 44, 57, 58, 59, 125

Ralph Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1997. 3, 4, 50

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60, 1992. 23

Matthias W. Seeger and Hannes Nickisch. Large scale variational inference and experimental design for sparse generalized linear models. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

Matthias W. Seeger and Hannes Nickish. Large scale bayesian inference and experimental design for sparse linear models. *SIAM Journal on Imaging Sciences*, 4:166–199, 2011. 79, 89

Matthias W. Seeger, Hannes Nickish, Rolf Pohmann, and Bernhard Schölkopf. Bayesian experimental design of magnetic resonance imaging sequences. *Neural Information Processing Systems*, 21:1441–1448, 2009a. 25

Matthias W. Seeger, Hannes Nickish, Rolf Pohmann, and Bernhard Schölkopf. Optimization of k-space trajectiories for compressed sensing by bayesian experimental design. *Magnetic Resonance in Medicine*, 63:116–126, 2009b. 2, 11, 25, 77, 78, 79, 89

Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

David Taubman and Michael Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. The Kluwer International Series in Engineering and Computer Science, 2002. 26

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

Wotao Yin, Stanley Osher, Donald Goldfarb, and Jérôme Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1:143–168, 2008. 4, 5, 60

Eitan Yudilevich and Henry Stark. Spiral sampling: Theory and an application to magnetic resonance imaging. *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, 5(4):542–553, 1988. 97

# List of Figures

Appendix A

# Notation

| | |
|---|---|
| $\alpha$, $t$, ... | scalars |
| $\boldsymbol{u}$, $\boldsymbol{v}$, ... | vectors |
| $i$ | the imaginary unit |
| $\alpha g$ | defines function $f(\boldsymbol{x}) := \alpha g(\boldsymbol{x})$ |
| $\|\| \cdot \|\|_\varepsilon$ | is the differentiable surrogate of the $l_1$-norm |
| $\|\|\boldsymbol{A}\|\|_{l_2}$ | stands for the operator norm; $\|\|\boldsymbol{A}\|\|_{l_2} = \sup_{\|\|\boldsymbol{u}\|\|_{l_2}=1} \|\|\boldsymbol{A}\boldsymbol{u}\|\|_{l_2}$ |
| $\equiv_{\boldsymbol{a}}$ | equality modulo terms independent of $\boldsymbol{a}$; $t_1(\boldsymbol{a}) + t_2(\bar{\boldsymbol{c}}) \equiv_{\boldsymbol{a}} t_1(\boldsymbol{a})$ |
| $\equiv_{\boldsymbol{a}:=\boldsymbol{b}}$ | substitution of all occurrences of $\boldsymbol{b}$ in the lhs by $\boldsymbol{a}$ in the rhs |
| $\cdot$ | denotes the scalar multiplication |
| $\otimes$ | denotes the Kronecker product |
| $\odot$ | denotes the component-wise multiplication |
| $\overline{\boldsymbol{v}}$ | denotes the complex conjugate |
| $(x)_+$ | denotes $\max\{x, 0\}$ |
| $\mathbb{R}$ | denotes set of real numbers |
| $\overline{\mathbb{R}}$ | denotes set of extended real numbers, that is $\overline{\mathbb{R}} := [-\infty, \infty]$ |
| $\mathbb{R}_{++}$ | denotes set of positive real numbers |
| $\mathbb{R}_{+}$ | denotes $\mathbb{R}_{++} \cup \{0\}$ |
| $\mathbb{C}$ | denotes set of complex numbers |
| $\mathbb{N}$ | denotes set of natural numbers |
| $\mathbb{Z}$ | denotes set of integers |
| $\mathcal{C}$ | denotes real-valued surrogate of the complex field $\mathbb{C}$, formally $\mathcal{C} := \mathbb{R}^2$ |
| $\mathcal{P}$ | denotes P-set |
| $\delta_{a\in\mathcal{A}}$ | indicator function of the set $\mathcal{A}$ |
| $\boldsymbol{I}_n$ | denotes $n$-by-$n$ identity matrix |
| $\boldsymbol{B}_{(a)}$ | denotes the wavelet transform |
| $\boldsymbol{B}_{(r)}$ | denotes the differential operator |
| $\boldsymbol{B}_{(i)}$ | denotes the imaginary part penalizing operator |
| $\boldsymbol{F}$ | denotes the Fourier transform |
| $\boldsymbol{X}$ | denotes the subsampling Fourier operator |
| $\mathcal{J}$ | denotes set of the chosen indices |
| $\boldsymbol{I}_{\mathcal{J},\cdot}$ | denotes the subsampling operator |
| $\boldsymbol{I}_{\cdot,\mathcal{J}}$ | denotes the upsampling operator; $\boldsymbol{I}_{\cdot,\mathcal{J}} = \boldsymbol{I}_{\mathcal{J},\cdot}^T$ |
| $[a_i]_i$ | vector that contains values $a_i$ from some set of indices $\mathcal{I}$ |
| $\boldsymbol{v}^{\mathcal{M}}$ | for a given $\boldsymbol{v}$ the corresponding matrix is denoted as $\boldsymbol{v}^{\mathcal{M}}$ |
| $n_x$ | number of columns in the image |
| $n_y$ | number of rows in the image |

Appendix B

# A Few Words about Notation

---

Assume that $\mathcal{I}$ is a finite subset of the set of natural numbers $\mathbb{N}$ with cardinality $m$. For a given vector $\boldsymbol{v} \in \mathbb{R}^n$, where $n \geq m$, the notation $\boldsymbol{v}_{\mathcal{I}}$ stands for

$$\boldsymbol{v}_{\mathcal{I}} := [v_i]_{i \in \mathcal{I}} \quad \text{where } u_i \in \mathbb{R}$$

In similar way if $\boldsymbol{u} \in \mathcal{C}^n$, where $n \geq m$, we have

$$\boldsymbol{u}_{\mathcal{I}} := [\boldsymbol{u}_i]_{i \in \mathcal{I}} \quad \text{where } \boldsymbol{u}_i \in \mathcal{C}$$

Assume that a vector $\boldsymbol{v} \in \mathbb{R}^n$ is given. If it is known that $\boldsymbol{v}$ was obtained from a matrix $\boldsymbol{A} \in \mathbb{R}^{n_y \times n_x}$ $(n = n_x n_y)$ by the matlab-like reshape operation, that is $\boldsymbol{v} := \text{reshape}(\boldsymbol{A}, n_x n_y, 1)$, then $\boldsymbol{v}^{\mathcal{M}} := \boldsymbol{A}$.

For each $z = a + ib \in \mathbb{C}$ let $\nu : \mathbb{C} \to \mathcal{C}$ be a mapping, between complex-set $\mathbb{C}$ and its real-valued surrogate $\mathcal{C}$, defined to be $\nu(z) := \begin{bmatrix} a \\ b \end{bmatrix}$. We can consider the following extension of $\nu$ for an arbitrary vector $\boldsymbol{z} \in \mathbb{C}^n$

$$\nu(\boldsymbol{z}) := [\nu(z_j)]_{j=1}^{n}$$

Assume now that a vector $\boldsymbol{u} \in \mathcal{C}^n$ is given. The vector $\boldsymbol{u}$ corresponds to the complex-valued vector $\boldsymbol{z}$ such that $\nu(\boldsymbol{z}) = \boldsymbol{u}$. If it is known that $\boldsymbol{z}$ was obtained from a matrix $\boldsymbol{Z} \in \mathbb{C}^{n_y \times n_x}$ $(n = n_x n_y)$ by the matlab-like reshape operation then $\boldsymbol{u}^{\mathcal{M}} := \boldsymbol{Z}$. To conclude, the notation $\boldsymbol{u}^{\mathcal{M}}$ denotes the matrix form of the vector $\boldsymbol{u}$.

In similar way, for fixed matrix $\boldsymbol{B}$, by $\boldsymbol{B}^{\mathcal{M}}$ we mean the linear operator that corresponds to $\boldsymbol{B}$ but works with matrix $\boldsymbol{u}^{\mathcal{M}}$. More precisely, if for every $\boldsymbol{u}$ and every $\boldsymbol{v}$ there exist a linear operator $\boldsymbol{P}$ such that $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u}$ if and only if $\boldsymbol{v}^{\mathcal{M}} = \boldsymbol{P}\boldsymbol{u}^{\mathcal{M}}$, then $\boldsymbol{B}^{\mathcal{M}} := \boldsymbol{P}$. Sometimes we also use $(\boldsymbol{A}\boldsymbol{B})^{\mathcal{M}}$ to denote $\boldsymbol{A}^{\mathcal{M}}\boldsymbol{B}^{\mathcal{M}}$.

We also use column-storage representation of matrices. Using matlab-like notation it corresponds to $\boldsymbol{u} := \mathit{reshape}(\boldsymbol{u}^{\mathcal{M}}, n_x n_y, 1)$.

Appendix C

# Mathematical Background

---

**Definition C.1** (Linear Operator). *Let $T : \mathbb{R}^n \to \mathbb{R}^m$ be an operator. Then $T(\cdot)$ is linear if and only if, for every $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$, the following holds*

$$\forall_{\alpha,\beta \in \mathbb{R}}\ T(\alpha \boldsymbol{v} + \beta \boldsymbol{u}) = \alpha T(\boldsymbol{v}) + \beta T(\boldsymbol{u})$$

**Definition C.2** (Projection). *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be a linear operator. Then $T(\cdot)$ is a projection if and only if, for every $\boldsymbol{u} \in \mathbb{R}^n$, the following holds*

$$T(T(\boldsymbol{u})) = T(\boldsymbol{u})$$

**Definition C.3** (Convex Set). *A set $\mathcal{C}$ is convex if and only if for every $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$ and every $0 \leq \alpha \leq 1$, we have*

$$\alpha \boldsymbol{c}_1 + (1 - \alpha)\boldsymbol{c}_2 \in \mathcal{C}$$

**Definition C.4** (Extended-valued Function). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be any function. Its extended-valued version is defined as*

$$\hat{f}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in dom(f) \\ \infty & \text{otherwise} \end{cases}$$

*In this thesis we always assume extended-valued versions of functions.*

**Definition C.5** (Proper Function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is proper if and only if there exists $\boldsymbol{x}$ such that $f(\boldsymbol{x}) < \infty$ and for every $\boldsymbol{y}$ the following $f(\boldsymbol{y}) > -\infty$ holds.*

**Definition C.6** (Coercive Function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is coercive if and only if $f(\boldsymbol{x}) \to \infty$ as $||\boldsymbol{x}|| \to \infty$.*

**Definition C.7** (Convex Function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its domain $dom(f)$ is a convex set and for every $\boldsymbol{x}, \boldsymbol{y} \in dom(f)$ and every $0 \leq \alpha \leq 1$, we have*

$$f(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha) f(\boldsymbol{y})$$

**Definition C.8** (Operator-norm). *Let $\boldsymbol{A} : \mathbb{R}^{m \times n}$ be a matrix. Then the operator-norm is defined as*

$$||\boldsymbol{A}||_{l_2} := \sup_{\boldsymbol{v} \in \mathbb{R}^n, ||\boldsymbol{v}||_{l_2} = 1} ||\boldsymbol{A}\boldsymbol{v}||_{l_2}$$

**Definition C.9** (p-norm). *Let $V := \mathbb{C}^n$ be a complex-valued vector space. Then the $l_p$-norm is defined as*

$$\forall_{\boldsymbol{u} \in \mathbb{C}^n}\ ||\boldsymbol{u}||_p := \left( \sum_{j=0}^{n-1} |u_i|^p \right)^{\frac{1}{p}}$$

*In this thesis two important special cases of the $l_p$-norm are considered: the $l_1$-norm and the $l_2$-norm.*

**Definition C.10** (Dual Norm)**.** *Let $V := \mathbb{R}^n$ be a vector space endowed with a norm $||\cdot||$. Then the dual norm $||\cdot||^*$ is defined as*

$$\forall_{\boldsymbol{v}\in\mathbb{R}^n} \quad ||\boldsymbol{v}||^* := \sup_{\boldsymbol{u}\in\mathbb{R}^n} \{\langle \boldsymbol{u}, \boldsymbol{v}\rangle \mid ||\boldsymbol{u}|| \leq 1\}$$

**Definition C.11** (Convex Optimization Problem)**.** *Convex optimization problem in standard form is stated as*

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{u}} & f(\boldsymbol{u}) \\ s.t. & \boldsymbol{X}\boldsymbol{u} = \boldsymbol{y} \\ & h_j(\boldsymbol{u}) \leq 0 \quad j \in \mathcal{J} \end{array} \tag{C.1}$$

*where $f$ and $h_j$ are convex functions for all $j \in \mathcal{J}$. In addition, we say that a point $\boldsymbol{u}$ is feasible if all equality and inequality constraints are satisfied at $\boldsymbol{u}$. Moreover, a feasible point $\boldsymbol{u}^\star$ is called a solution (or a minimizer) of problem (C.1) if and only if*

$$f(\boldsymbol{u}^\star) \leq f(\boldsymbol{u}) \text{ for every feasible } \boldsymbol{u}$$

**Definition C.12** (Lagrangian)**.** *The so called Lagrange function associated to the problem C.1 is*

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) = f(\boldsymbol{u}) + \boldsymbol{\lambda}^T(\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}) + \sum_{j\in\mathcal{J}} \gamma_j h_j(\boldsymbol{u}) \tag{C.2}$$

*where $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ are vectors of Lagrange multipliers.*

**Definition C.13** (Lagrange Dual Problem)**.** *Let $\mathcal{L}$ be Lagrange function (C.2). Define the so called Lagrange dual function as follows*

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \underset{\boldsymbol{u}}{\text{minimize}} \ \mathcal{L}(\boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) \tag{C.3}$$

*Then Lagrange dual associated to problem C.1 is*

$$\begin{array}{ll} \text{maximize}_{\boldsymbol{\lambda},\boldsymbol{\gamma}} & g(\boldsymbol{\lambda}, \boldsymbol{\gamma}) \\ s.t. & \boldsymbol{\lambda} \succeq \boldsymbol{0} \end{array} \tag{C.4}$$

*where $\succeq$ denotes the component wise inequality $\geq$. Then problem (C.1) is called the primal problem and problem (C.4) is called the dual problem. Moreover, every minimizer of the primal problem is called a primal optimal, and every maximizer of the dual problem is called a dual optimal.*

**Definition C.14** (KKT Conditions for Convex Problems)**.** *Consider the convex optimization problem (C.1). Let $\overline{\boldsymbol{u}}$, $\overline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\mu}}$ be any points that satisfies the KKT conditions*

$$\begin{array}{lll} h_j(\overline{\boldsymbol{u}}) & \leq 0 & j \in \mathcal{J} \\ \boldsymbol{X}\overline{\boldsymbol{u}} & = \boldsymbol{y} & \\ \overline{\boldsymbol{\lambda}} & \succeq \boldsymbol{0} & \\ \overline{\boldsymbol{\lambda}}^T \boldsymbol{h}(\overline{\boldsymbol{u}}) & = 0 & \\ \nabla_{\boldsymbol{u}} \mathcal{L}(\overline{\boldsymbol{u}}, \overline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\mu}}) & = 0 & \end{array}$$

*where $\boldsymbol{h}(\overline{\boldsymbol{u}}) := [h_j(\overline{\boldsymbol{u}})]_j$. Then $\overline{\boldsymbol{u}}$, $\overline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\mu}}$ are primal and dual optimal.*

**Definition C.15** (Equivalent Problems, Informally)**.** *Following Boyd and Vandenberghe [2004], we call two problems equivalent if from the solution of the first problem, the solution of the second problem can be readily found.*

Appendix D

# Supplementary Proofs

## D.1 Fourier Transform

Let recall definition of the 2-D discrete Fourier transform $\boldsymbol{F}^{\mathcal{M}}$

$$(\boldsymbol{F}\boldsymbol{u})^{\mathcal{M}}_{\gamma,\omega} := \frac{1}{\sqrt{n}} \sum_{x=0}^{n_x-1} \sum_{y=0}^{n_y-1} \boldsymbol{u}^{\mathcal{M}}_{y,x} \exp\left(-i2\pi\left(\frac{x\cdot\omega}{n_x} + \frac{y\cdot\gamma}{n_y}\right)\right) \tag{D.1}$$

and its inverse

$$(\boldsymbol{F}^H\ \boldsymbol{u})^{\mathcal{M}}_{y,x} := \frac{1}{\sqrt{n}} \sum_{\omega=0}^{n_x-1} \sum_{\gamma=0}^{n_y-1} \boldsymbol{u}^{\mathcal{M}}_{\gamma,\omega} \exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x} + \frac{y\cdot\gamma}{n_y}\right)\right) \tag{D.2}$$

where $n := n_x \cdot n_y$.

In order to prove that $(\boldsymbol{F}^H)^{\mathcal{M}}$ is actually the inverse transform to $\boldsymbol{F}^{\mathcal{M}}$ we have to show that the following holds

$$(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}} = (\boldsymbol{F}^H(\boldsymbol{F}\boldsymbol{u}))^{\mathcal{M}} = \boldsymbol{u}^{\mathcal{M}}$$

Let start from the following lemma

**Lemma D.1.** *Let fix $M \in \mathbb{N} - \{0\}$ and define*

$$p_k := \exp\left(i2\pi\frac{k}{M}\right)$$

*where $k \in \{0, 1, \ldots, M-1\}$. Then the following holds*

$$\sum_{m=0}^{M-1} p_k^m = \begin{cases} M & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

*Proof.* We will consider two cases:

- If $k = 0$ then $p_k = 1$ and therefore

$$\sum_{m=0}^{M-1} p_k^m = \sum_{m=0}^{M-1} 1 = M$$

  This ends the first part of the proof.

- Assume now that $k \neq 0$. Since $p_k \neq 1$ we can use geometric series formula to obtain

$$\sum_{m=0}^{M-1} p_k^m = \frac{1 - p_k^M}{1 - p_k}$$

Because

$$p_k^M = \left(\exp\left(i2\pi\frac{k}{M}\right)\right)^M = \exp\left(i2\pi\frac{k\cdot M}{M}\right) = \exp\left(i2\pi k\right) = 1$$

we have

$$\sum_{m=0}^{M-1} p_k^m = \frac{1-p_k^M}{1-p_k} = 0$$

This ends the last part of the proof.

□

Now we can prove the main theorem

**Theorem D.1.** *The following equality holds*

$$(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}} = (\boldsymbol{F}^H(\boldsymbol{F}\boldsymbol{u}))^{\mathcal{M}} = \boldsymbol{u}^{\mathcal{M}}$$

*Proof.* We will prove this theorem in two steps:

- First we show that $(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}} = \boldsymbol{u}^{\mathcal{M}}$ holds. For every $\gamma$ and $\omega$ we have

$$\begin{aligned}
&(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}}_{\gamma,\omega} \\
&= \frac{1}{\sqrt{n}}\sum_{x=0}^{n_x-1}\sum_{y=0}^{n_y-1}(\boldsymbol{F}^H\boldsymbol{u})^{\mathcal{M}}_{y,x}\exp\left(-i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \frac{1}{n}\sum_{x=0}^{n_x-1}\sum_{y=0}^{n_y-1}\left(\sum_{\eta=0}^{n_x-1}\sum_{\zeta=0}^{n_y-1}\boldsymbol{u}^{\mathcal{M}}_{\zeta,\eta}\exp\left(i2\pi\left(\frac{x\cdot\eta}{n_x}+\frac{y\cdot\zeta}{n_y}\right)\right)\right)\exp\left(-i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \frac{1}{n}\sum_{\eta=0}^{n_x-1}\sum_{\zeta=0}^{n_y-1}\boldsymbol{u}^{\mathcal{M}}_{\zeta,\eta}\sum_{x=0}^{n_x-1}\exp\left(\frac{i2\pi x}{n_x}(\eta-\omega)\right)\sum_{y=0}^{n_y-1}\exp\left(\frac{i2\pi y}{n_y}(\zeta-\gamma)\right) \\
&= \frac{1}{n}\sum_{\eta=0}^{n_x-1}\sum_{\zeta=0}^{n_y-1}\boldsymbol{u}^{\mathcal{M}}_{\zeta,\eta}\sum_{x=0}^{n_x-1}\left(\exp\left(\frac{i2\pi}{n_x}(\eta-\omega)\right)\right)^x\sum_{y=0}^{n_y-1}\left(\exp\left(\frac{i2\pi}{n_y}(\zeta-\gamma)\right)\right)^y \\
&= \boldsymbol{u}^{\mathcal{M}}_{\gamma,\omega}
\end{aligned}$$

  where we used Lemma D.1 to transit from the second last row to the last row.

- Similarly, for every $\gamma$ and $\omega$ we have

$$\begin{aligned}
&(\boldsymbol{F}^H(\boldsymbol{F}\boldsymbol{u}))^{\mathcal{M}}_{y,x} \\
&= \frac{1}{\sqrt{n}}\sum_{\omega=0}^{n_x-1}\sum_{\gamma=0}^{n_y-1}(\boldsymbol{F}\boldsymbol{u})^{\mathcal{M}}_{\gamma,\omega}\exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \frac{1}{n}\sum_{\omega=0}^{n_x-1}\sum_{\gamma=0}^{n_y-1}\left(\sum_{a=0}^{n_x-1}\sum_{b=0}^{n_y-1}\boldsymbol{u}^{\mathcal{M}}_{b,a}\exp\left(-i2\pi\left(\frac{a\cdot\omega}{n_x}+\frac{b\cdot\gamma}{n_y}\right)\right)\right)\exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \sum_{a=0}^{n_x-1}\sum_{b=0}^{n_y-1}\boldsymbol{u}^{\mathcal{M}}_{b,a}\sum_{\omega=0}^{n_x-1}\left(\exp\left(\frac{i2\pi}{n_x}(x-a)\right)\right)^\omega\sum_{\gamma=0}^{n_y-1}\left(\exp\left(\frac{i2\pi}{n_y}(y-b)\right)\right)^\gamma \\
&= \boldsymbol{u}^{\mathcal{M}}_{y,x}
\end{aligned}$$

  where we again used Lemma D.1 to transit from the second last row to the last row.

Therefore we can conclude that the following holds

$$(\boldsymbol{F}(\boldsymbol{F}^H\boldsymbol{u}))^{\mathcal{M}} = (\boldsymbol{F}^H(\boldsymbol{F}\boldsymbol{u}))^{\mathcal{M}} = \boldsymbol{u}^{\mathcal{M}}$$

□

We can also relate the discrete Fourier transform applied to the complex conjugate of some vector with the inverse discrete Fourier transform applied to the complex conjugate of the reverse of the vector. The lemma is as follows

**Lemma D.2.** *Let $\boldsymbol{F}$ be the 2-D discrete Fourier transform and $(\cdot)_R$ the reverse operation defined by (4.15). Then the following holds*

$$\forall_{\boldsymbol{v}\in\mathbb{C}^n}\ \overline{\boldsymbol{F}^H\boldsymbol{v}} = \boldsymbol{F}\overline{\boldsymbol{v}} = \boldsymbol{F}^H\overline{\boldsymbol{v}_R}$$

*Proof.* Fix $\boldsymbol{v} \in \mathbb{C}^{n=n_y\cdot n_x}$. Then we have

$$\begin{aligned}
&\sqrt{n}(\boldsymbol{F}^H\overline{\boldsymbol{v}_R})_{(y,x)} \\
&= \sum_{\gamma=0}^{n_y-1}\sum_{\omega=0}^{n_x-1}(\overline{\boldsymbol{v}_R})_{(\gamma,\omega)}\exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \overline{\boldsymbol{v}_{(0,0)}} + \sum_{\omega=1}^{n_x-1}\overline{\boldsymbol{v}_{(0,n_x-\omega)}}\exp\left(i2\pi\frac{x\cdot\omega}{n_x}\right) + \sum_{\gamma=1}^{n_y-1}\overline{\boldsymbol{v}_{(n_y-\gamma,0)}}\exp\left(i2\pi\frac{y\cdot\gamma}{n_y}\right) \\
&\quad + \sum_{\omega=1}^{n_x-1}\sum_{\gamma=1}^{n_y-1}\overline{\boldsymbol{v}_{(n_y-\gamma,n_x-\omega)}}\exp\left(i2\pi\left(\frac{x\cdot\omega}{n_x}+\frac{y\cdot\gamma}{n_y}\right)\right) \\
&= \overline{\boldsymbol{v}_{(0,0)}} + \sum_{a=1}^{n_x-1}\overline{\boldsymbol{v}_{(0,a)}}\exp\left(-i2\pi\frac{x\cdot a}{n_x}\right) + \sum_{b=1}^{n_y-1}\overline{\boldsymbol{v}_{(b,0)}}\exp\left(-i2\pi\frac{y\cdot b}{n_y}\right) \\
&\quad + \sum_{a=1}^{n_x-1}\sum_{b=1}^{n_y-1}\overline{\boldsymbol{v}_{(b,a)}}\exp\left(-i2\pi\left(\frac{x\cdot a}{n_x}+\frac{y\cdot b}{n_y}\right)\right) \\
&= \sqrt{n}(\boldsymbol{F}\overline{\boldsymbol{v}})_{(y,x)} = \sqrt{n}(\overline{\boldsymbol{F}^H\boldsymbol{v}})_{(y,x)}
\end{aligned}$$

□

For additional information about the discrete Fourier transform, see the Mallat's book Mallat [2008].

## D.2 The Reverse Operation

Recall that the reverse operation was defined as follows

$$\boldsymbol{v}_R^{\mathcal{M}} := [v_{[\boldsymbol{n}-\boldsymbol{k}]_{\boldsymbol{n}}}]_{\boldsymbol{k}=(0,0)}^{\boldsymbol{n}-(1,1)} \tag{D.3}$$

Now, we prove two lemmas used in Section 4.2.

**Lemma D.3.** *Let $(\cdot)_R$ be the reverse operation given by (D.3). Then the following holds*

$$(\boldsymbol{v}_R)_R = \boldsymbol{v}$$

*Proof.* Let $\boldsymbol{v} \in \mathbb{C}^n$. Then we have

$$\boldsymbol{v}_R = [v_{[\boldsymbol{n}-\boldsymbol{k}]_{\boldsymbol{n}}}]_{\boldsymbol{k}=(0,0)}^{\boldsymbol{n}-(1,1)}$$

and

$$(\boldsymbol{v}_R)_R = \left([v_{[\boldsymbol{n}-\boldsymbol{k}]_{\boldsymbol{n}}}]_{\boldsymbol{k}=(0,0)}^{\boldsymbol{n}-(1,1)}\right)_R$$

Therefore

$$(\boldsymbol{v}_R)_R = [v_{[\boldsymbol{n}-[\boldsymbol{n}-\boldsymbol{k}]_{\boldsymbol{n}}]_{\boldsymbol{n}}}]_{\boldsymbol{k}=(0,0)}^{\boldsymbol{n}-(1,1)}$$

Assume now that $\boldsymbol{k} > (0,0)$. Then we have $\boldsymbol{n} - \boldsymbol{k} < \boldsymbol{n}$ and

$$[v_{[\boldsymbol{n}-\boldsymbol{n}+\boldsymbol{k}]_{\boldsymbol{n}}}]_{\boldsymbol{k}=(1,1)}^{\boldsymbol{n}-(1,1)} = [v_{\boldsymbol{k}}]_{\boldsymbol{k}=(1,1)}^{\boldsymbol{n}-(1,1)} \tag{D.4}$$

Assume now that $\boldsymbol{k} = (0,x)$ but $x > 0$. Then

$$v_{[\boldsymbol{n}-[\boldsymbol{n}-(0,x)]_{\boldsymbol{n}}]_{\boldsymbol{n}}} = v_{[(n_y,n_x)-[(n_y,n_x-x)]_{\boldsymbol{n}}]_{\boldsymbol{n}}} = v_{[(n_y,n_x)-(0,n_x-x)]_{\boldsymbol{n}}} = v_{(0,x)} \tag{D.5}$$

Proof for the case $\boldsymbol{k} = (y,0)$ and $y > 0$ is similar to the previous one. Finally, assume that $\boldsymbol{k} = (0,0)$. Then

$$v_{[\boldsymbol{n}-[\boldsymbol{n}-(0,0)]_{\boldsymbol{n}}]_{\boldsymbol{n}}} = v_{(0,0)} \tag{D.6}$$

Putting all together yields

$$(\boldsymbol{v}_R)_R = \boldsymbol{v}$$

This ends the proof. □

**Lemma D.4.** *The reverse operation is a linear operation.*

*Proof.* Let $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{C}^n$. Directly from the definition we can see that $(\alpha\boldsymbol{v} + \beta\boldsymbol{u})_R = \alpha(\boldsymbol{v})_R + \beta(\boldsymbol{u})_R$ holds. □

Appendix E

# Standard Vectors

Let $\boldsymbol{e}_{(j)} \in \mathbb{R}^n$ be the vector with 1 at $j$-th position and 0 everywhere else. Formally

$$(\boldsymbol{e}_{(j)})_l := \begin{cases} 1 & \text{if } l = j \\ 0 & \text{otherwise} \end{cases}$$

We call the vector $\boldsymbol{e}_{(j)}$ the $j$-th standard vector. Notice that the $j$-th standard vector can be used as a selector which returns $j$-th entry of another vector. That is

$$\boldsymbol{e}_{(j)}^T \boldsymbol{v} = v_j \tag{E.1}$$

for an arbitrary $\boldsymbol{v} \in \mathbb{R}^n$. On the other hand the standard vector $\boldsymbol{e}_{(j)}$ can also be used to transform a scalar into a vector in $\mathbb{R}^n$. That is

$$\boldsymbol{e}_{(j)} v_j = \hat{\boldsymbol{v}}^{(j)} \tag{E.2}$$

where $v_j \in \mathbb{R}$ and

$$\hat{\boldsymbol{v}}_l^{(j)} := \begin{cases} v_j & \text{if } l = j \\ 0 & \text{otherwise} \end{cases}$$

Note that also the following holds

$$\sum_{j=1}^{n} \boldsymbol{e}_{(j)} \boldsymbol{e}_{(j)}^T = \boldsymbol{I}$$

where $\boldsymbol{I} \in \mathbb{R}^n$ is the identity matrix.

Our definition of the standard vector can readily be extended to $\mathcal{C}^n$ by embedding

$$\boldsymbol{e}_{(j)}^C := \boldsymbol{e}_{(j)} \otimes \boldsymbol{I}_2$$

We call the vector $\boldsymbol{e}_{(j)}^C$ the $j$-th standard vector in $\mathcal{C}^n$.

Note the analogous results for $\boldsymbol{e}_{(j)}^C$ to (E.1) and (E.2). That is

$$(\boldsymbol{e}_{(j)}^C)^T \boldsymbol{s} = \boldsymbol{s}_j \tag{E.3}$$

where $\boldsymbol{s} \in \mathcal{C}^n$ and $\boldsymbol{s}_j \in \mathcal{C}$. Also

$$\boldsymbol{e}_{(j)}^C \boldsymbol{s}_j = \hat{\boldsymbol{s}}^{(j)} \tag{E.4}$$

where $\boldsymbol{s}_j \in \mathcal{C}$ and

$$\hat{\boldsymbol{s}}_l^{(j)} := \begin{cases} \boldsymbol{s}_j & \text{if } l = j \\ [0,0]^T & \text{otherwise} \end{cases}$$

In this thesis, for the sake of brevity, we will skip superscript $C$ in $\boldsymbol{e}^C_{(j)}$ and will use the following notation $\boldsymbol{e}_{(j)}$. It should be clear from the context if this notation refers to the standard vector in $\mathbb{R}^n$ or in $\mathcal{C}^n$. Moreover, since we are not interested in the particular entries of the standard vector anymore, we also skip $(\cdot)$ in the subscript. Finally, we use notation $\boldsymbol{e}_j$ to denote the $j$-th standard vector.

Appendix F

# Derivatives

Derivatives used in this thesis come from Minka [2001] and are based on the infinitesimal vectors. Let consider the following equation

$$f(\boldsymbol{u} + d\boldsymbol{u}) = f(\boldsymbol{u}) + \boldsymbol{A}d\boldsymbol{u} + (\text{higher order terms})$$

The matrix $\boldsymbol{A}$ is the derivative and its transpose is the gradient of $f$, that is $\boldsymbol{A} = \nabla f(\boldsymbol{u})^T$. Let define the differential $df(\boldsymbol{u})$ to be the linear part of $f(\boldsymbol{u} + d\boldsymbol{u}) - f(\boldsymbol{u})$, that is $df(\boldsymbol{u}) = \boldsymbol{A}d\boldsymbol{u}$. Therefore, in order to compute derivative of an expression, it is enough to compute the differential and then massage the result into the canonical form. After this we can read off the derivative as the coefficient of $d\boldsymbol{u}$. Minka [2001] considered six canonical forms but in this thesis only the following one $df(\boldsymbol{u}) = \boldsymbol{a}d\boldsymbol{u}$ is needed.

Now, we can compute derivative and gradient of the expression

$$\frac{1}{2}||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2$$

for some real-valued matrix $\boldsymbol{X}$, a constant vector $\boldsymbol{y}$ and variable $\boldsymbol{u}$. We have

$$\begin{aligned}
& d\,||\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}||_{l_2}^2 \\
&= d\,(\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{u} - \boldsymbol{y}) \\
&= d\,\left(\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{u} + \boldsymbol{y}^T\boldsymbol{y}\right) \\
&= d\,(\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}) - 2d\,(\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{u}) \\
&= (d\,\boldsymbol{u}^T)\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} + \boldsymbol{u}^T(d\,\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}) - 2\boldsymbol{y}^T\boldsymbol{X}(d\,\boldsymbol{u}) \\
&= (d\,\boldsymbol{u})^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u} + \boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}(d\,\boldsymbol{u}) - 2\boldsymbol{y}^T\boldsymbol{X}(d\,\boldsymbol{u}) \\
&= 2\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}(d\,\boldsymbol{u}) - 2\boldsymbol{y}^T\boldsymbol{X}(d\,\boldsymbol{u}) \\
&= 2\left(\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X} - \boldsymbol{y}^T\boldsymbol{X}\right)(d\,\boldsymbol{u})
\end{aligned}$$

where we applied the following rules (here $\boldsymbol{A}$ is a constant and $\boldsymbol{X}$, $\boldsymbol{Y}$ are variables; see also Minka [2001])

$$d\,\boldsymbol{A} = 0 \tag{F.1}$$

$$d\,\boldsymbol{A}\boldsymbol{X} = \boldsymbol{A}(d\,\boldsymbol{X}) \tag{F.2}$$

$$d\,(\boldsymbol{X} + \boldsymbol{Y}) = d\,\boldsymbol{X} + d\,\boldsymbol{Y} \tag{F.3}$$

$$d\,(\boldsymbol{X}\boldsymbol{Y}) = (d\boldsymbol{X})\boldsymbol{Y} + \boldsymbol{X}(d\,\boldsymbol{Y}) \tag{F.4}$$

$$d\,\boldsymbol{X}^T = (d\,\boldsymbol{X})^T \tag{F.5}$$

Since we massaged the result into the canonical form we can read off the gradient of $\frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2$ as a transpose of the expression $\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{X}-\boldsymbol{y}^T\boldsymbol{X}$, that is

$$\nabla\frac{1}{2}||\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}||_{l_2}^2=\left(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}-\boldsymbol{X}^T\boldsymbol{y}\right)=\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{u}-\boldsymbol{y}\right)$$

Another useful example is computing the gradient of the following expression

$$\sum_{j=1}^{m}w_j||\boldsymbol{s}_j||_{l_2}^2 \tag{F.6}$$

where $\boldsymbol{u}\in\mathcal{C}^n$, $\boldsymbol{s}:=\boldsymbol{B}\boldsymbol{u}\in\mathcal{C}^m$ and $w_j$ are some real-valued weights for every $j\in\{1,2,\ldots,m\}$. Equipped with the standard vectors, expression (F.6) can be written as

$$\sum_{j=1}^{m}w_j||\boldsymbol{e}_j^T\boldsymbol{s}||_{l_2}^2$$

Since

$$\begin{aligned}
&d\sum_{j=1}^{m}w_j||\boldsymbol{s}_j||_{l_2}^2\\
&=\sum_{j=1}^{m}w_j d\,||\boldsymbol{e}_j^T\boldsymbol{s}||_{l_2}^2\\
&=2\sum_{j=1}^{m}w_j\boldsymbol{s}^T\boldsymbol{e}_j\boldsymbol{e}_j^T(d\,\boldsymbol{s})\\
&=2\boldsymbol{s}^T\left(\sum_{j=1}^{m}w_j\boldsymbol{e}_j\boldsymbol{e}_j^T\right)(d\,\boldsymbol{s})\\
&=2\boldsymbol{u}^T\boldsymbol{B}^T\left(\sum_{j=1}^{m}w_j\boldsymbol{e}_j\boldsymbol{e}_j^T\right)\boldsymbol{B}(d\boldsymbol{u})
\end{aligned}$$

the gradient is

$$\nabla\sum_{j=1}^{m}w_j||\boldsymbol{s}_j||_{l_2}^2=2\boldsymbol{B}^T\left(\boldsymbol{W}\otimes\boldsymbol{I}_2\right)\boldsymbol{B}\boldsymbol{u} \tag{F.7}$$

where $\boldsymbol{I}_2\in\mathbb{R}^{2\times2}$ is the identity matrix and $\boldsymbol{W}$ is the diagonal matrix with the following diagonal entries

$$\boldsymbol{W}_{j,j}:=w_j$$

More details about the differential calculus used in this thesis can be found in Minka [2001].

Appendix G

# Line-search Methods

Good overview of the line-search methods the reader can find in Nocedal and Wright [1999]. Here we present only these methods that were used in this thesis.

**Backtracking (Armijo Rule).** Let $f$ be a continuously differentiable function. Then we can use the following rule in order to find a suitable step-size $\alpha$

$$f(\boldsymbol{u} + \alpha \boldsymbol{d}) \leq f(\boldsymbol{u}) + \sigma \alpha \nabla f(\boldsymbol{u})^T \boldsymbol{d} \tag{G.1}$$

for some $\alpha > 0$ and $\sigma \in (0, 1)$. The following algorithm is an implementation of the Rule (G.1)

**Algorithm 22** Backtracking (Armijo Rule)

**Require:** $\alpha_1 > 0$, $\beta \in (0, 1)$, $\sigma \in (0, 1)$, $\boldsymbol{u}$, $\boldsymbol{d}$

1: **for** $k = 1, 2, \ldots$ **do**
2:  **if** $f(\boldsymbol{u} + \alpha_k \boldsymbol{d}) \leq f(\boldsymbol{u}) + \sigma \alpha_k \nabla f(\boldsymbol{u})^T \boldsymbol{d}$ **then**
3:  break the loop
4:  **else**
5:  $\alpha_{k+1} := \beta \alpha_k$
6:  **end if**
7: **end for**

Intuitively, this algorithm chooses the step-size $\alpha$ such that the sufficient decrease occurs. In particular, if the function $f$ is convex and $\boldsymbol{u}$ is not a minimizer then $\nabla f(\boldsymbol{u})^T \boldsymbol{d} = -\gamma < 0$ and Rule (G.1) becomes

$$f(\boldsymbol{u} + \alpha \boldsymbol{d}) - f(\boldsymbol{u}) \leq -\sigma \alpha \gamma < 0$$

for some $\gamma > 0$.

**Backtracking for Projection.** Assume that $f$ is a function with Lipschitz continuous gradient. Some algorithms such as *Fast Gradient Projection* (Algorithm 4 or Algorithm 16) relies on an upper bound of the Lipschitz constant. If this upper bound is unknown or it is too large to be useful in practice then the following backtracking rule can be embedded in the algorithm

**Algorithm 23** Backtracking for Projection

**Require:** $\alpha_1 > 0$, $\beta \in (0, 1)$, $\boldsymbol{u}$

1: **for** $k = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{u}_{(\alpha_k)} = P_{\mathcal{K}}(\boldsymbol{u} - \alpha_k \nabla f(\boldsymbol{u}))$
3: $\quad$ **if** $f(\boldsymbol{u}_{(\alpha_k)}) \leq f(\boldsymbol{u}) + \left\langle \nabla f(\boldsymbol{u}), \boldsymbol{u}_{(\alpha_k)} - \boldsymbol{u} \right\rangle + \frac{1}{2\alpha_k} ||\boldsymbol{u}_{(\alpha_k)} - \boldsymbol{u}||_{l_2}^2$ **then**
4: $\quad\quad$ break the loop
5: $\quad$ **else**
6: $\quad\quad$ $\alpha_{k+1} := \beta \alpha_k$
7: $\quad$ **end if**
8: **end for**

This algorithm is searching for $\alpha_k$ such that $\frac{1}{\alpha_k}$ is the upper bound of the Lipschitz constant.

Appendix H

# Supplementary Figures

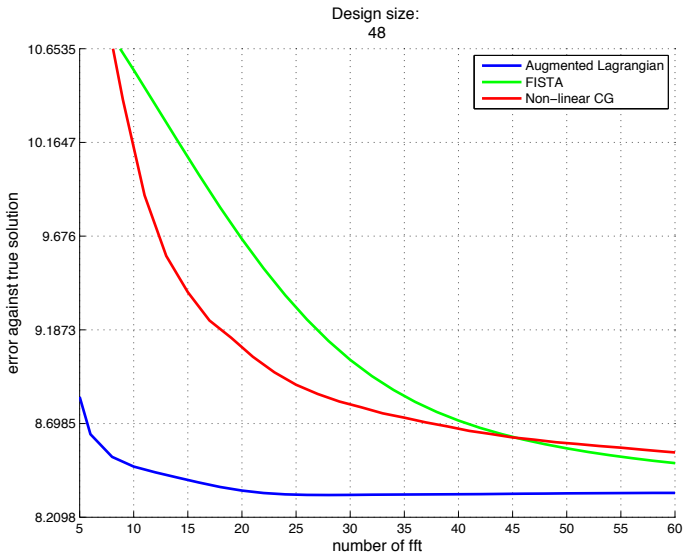



Figure H.1: Design size: 48. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

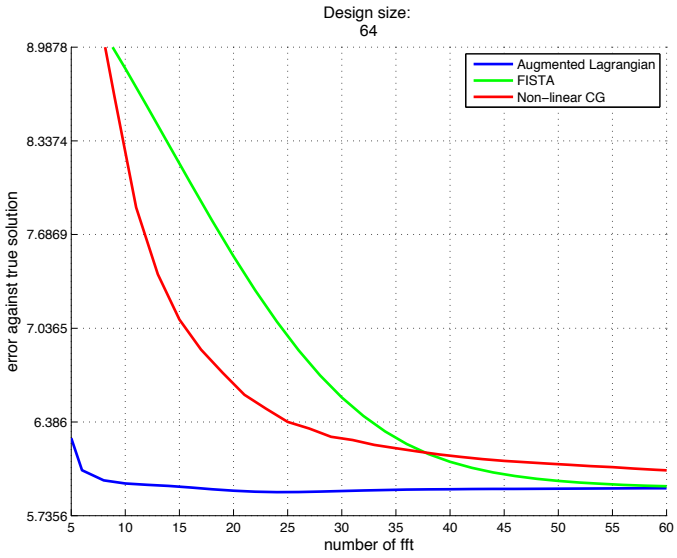



Figure H.2: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

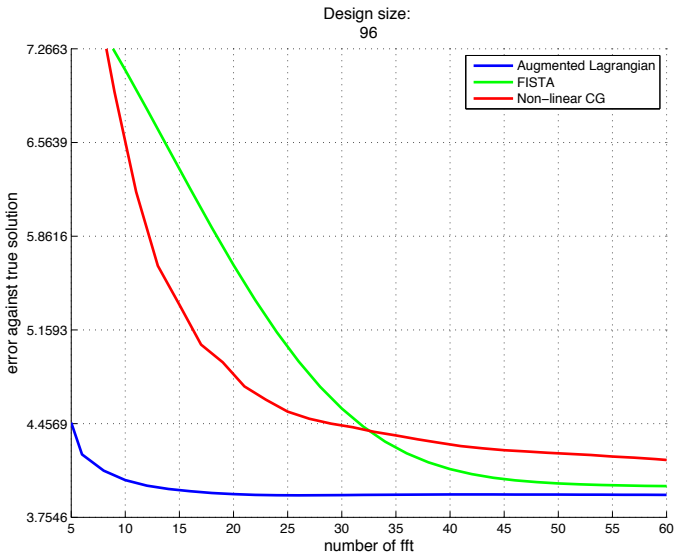



Figure H.3: Design size: 96. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

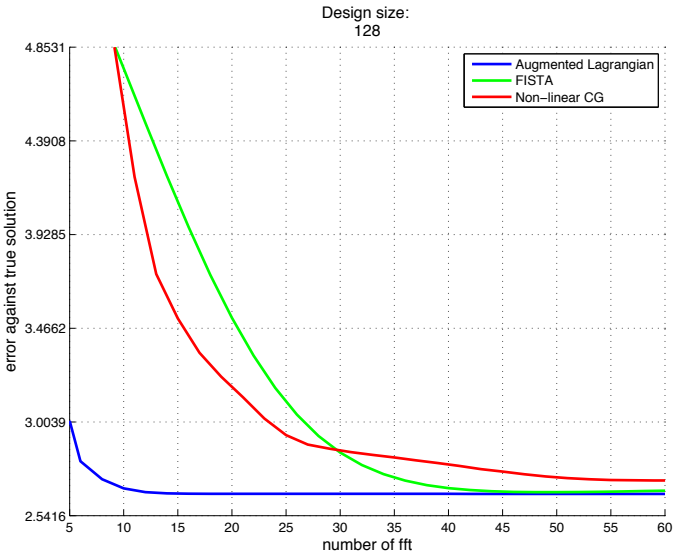



Figure H.4: Design size: 128. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

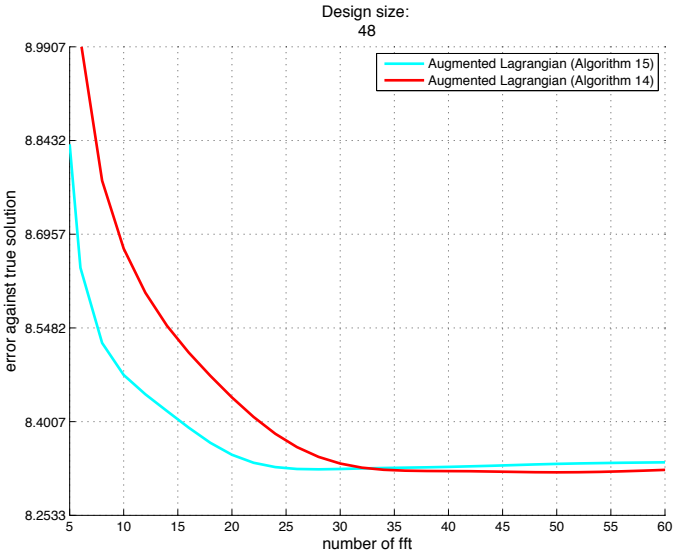



Figure H.5: Design size: 48. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

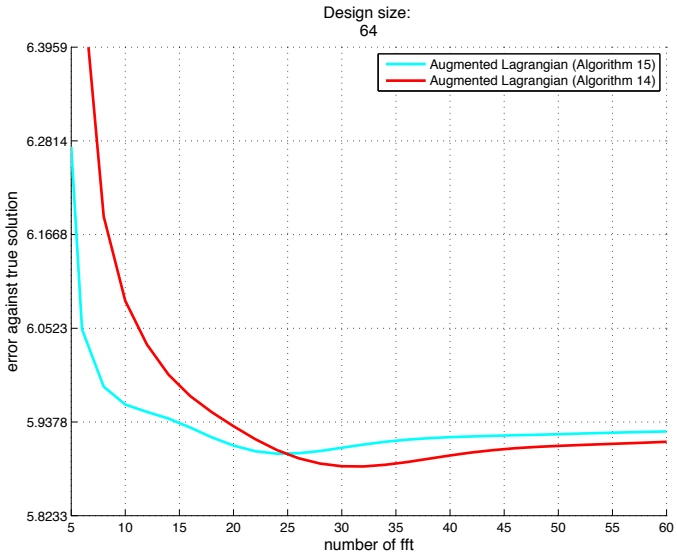



Figure H.6: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

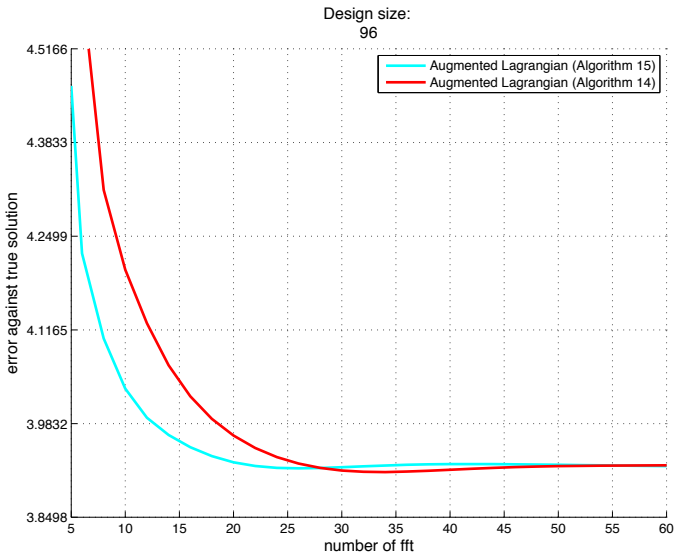



Figure H.7: Design size: 96. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).

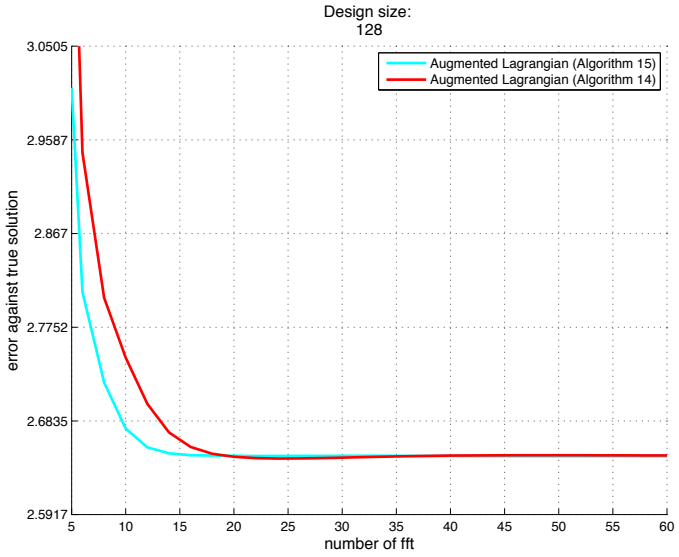
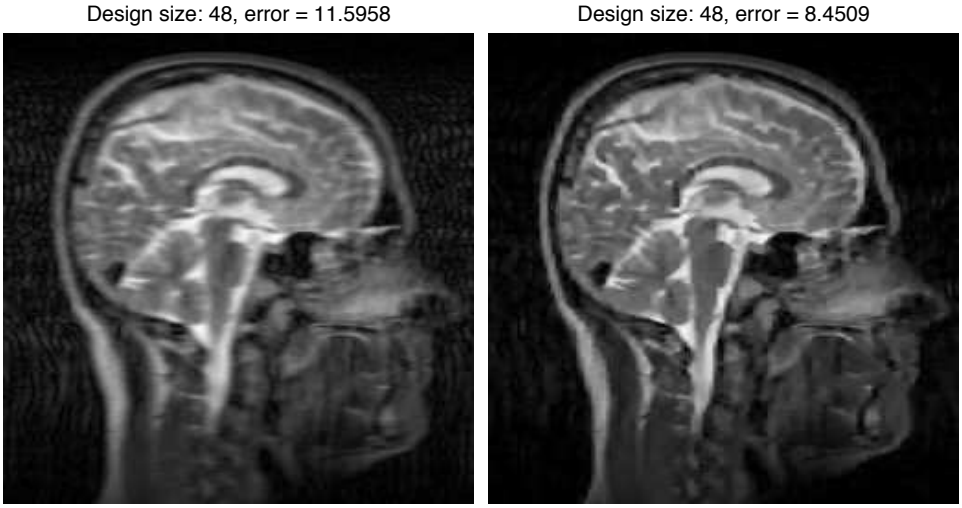
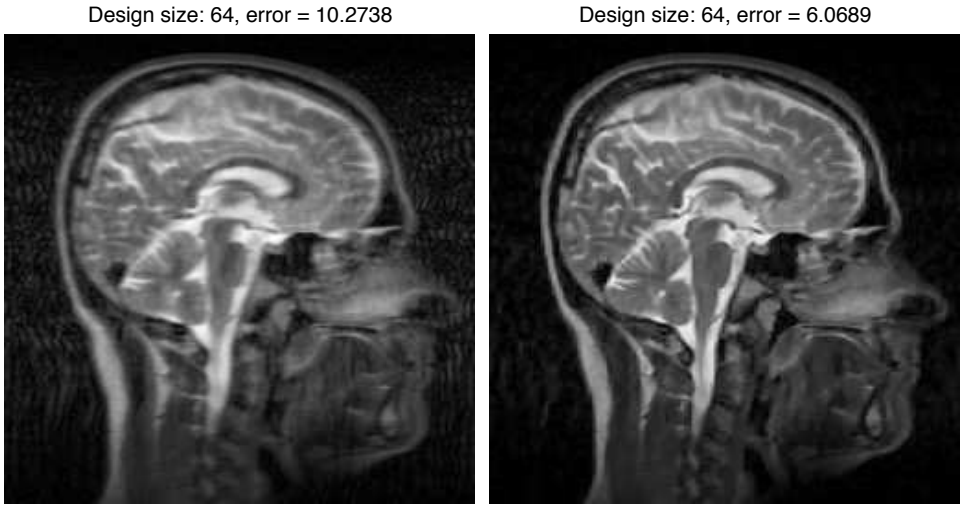



Figure H.8: Design size: 128. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects).



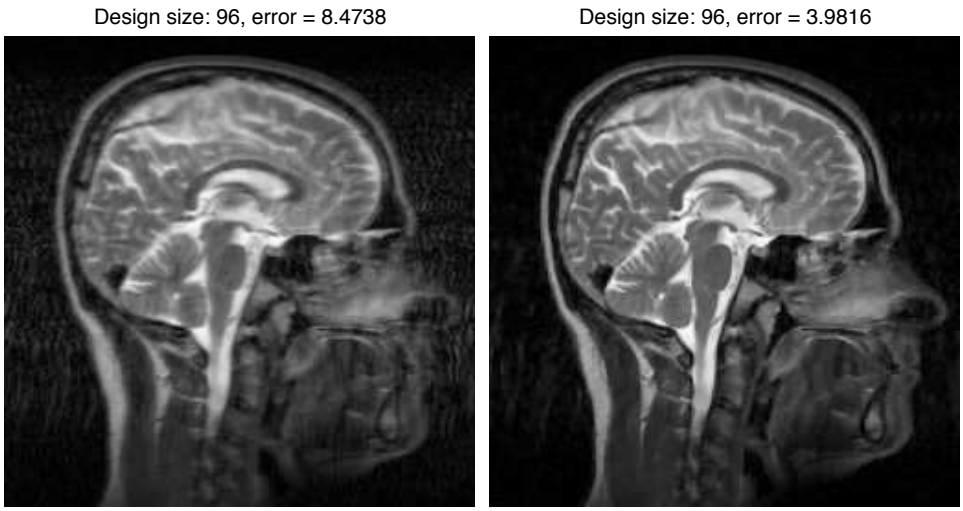

(a) Naive reconstruction (b) Reconstruction with model (3.7)

Figure H.9: Design size: 48. The reconstruction with model (3.7) was done by *Augmented Lagrangian*. The number of iterations was limited to 10 steps. Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.



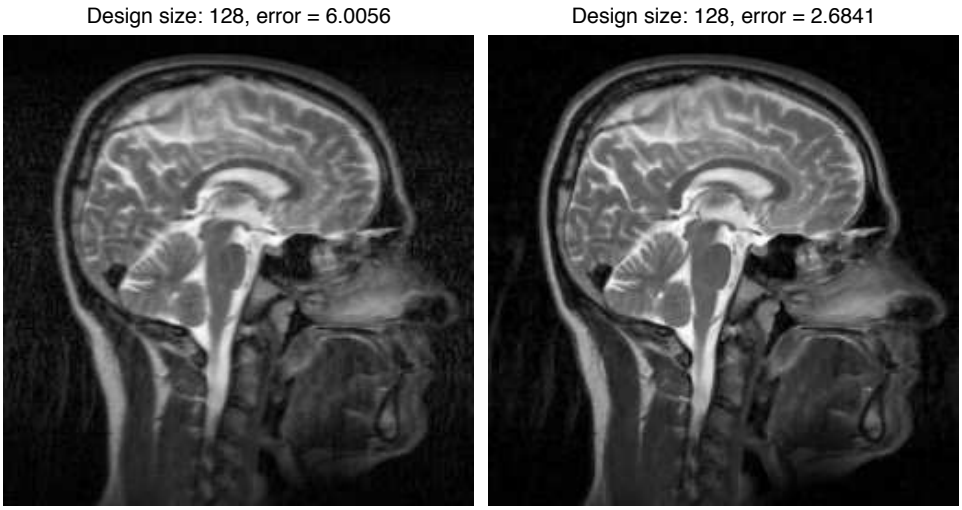

(a) Naive reconstruction (b) Reconstruction with model (3.7)

Figure H.10: Design size: 64. The reconstruction with model (3.7) was done by *Augmented Lagrangian*. The number of iterations was limited to 10 steps. Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Naive reconstruction (b) Reconstruction with model (3.7)

Figure H.11: Design size: 96. The reconstruction with model (3.7) was done by *Augmented Lagrangian*. The number of iterations was limited to 10 steps. Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Naive reconstruction (b) Reconstruction with model (3.7)

Figure H.12: Design size: 128. The reconstruction with model (3.7) was done by *Augmented Lagrangian*. The number of iterations was limited to 10 steps. Bayesian experimental design was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.

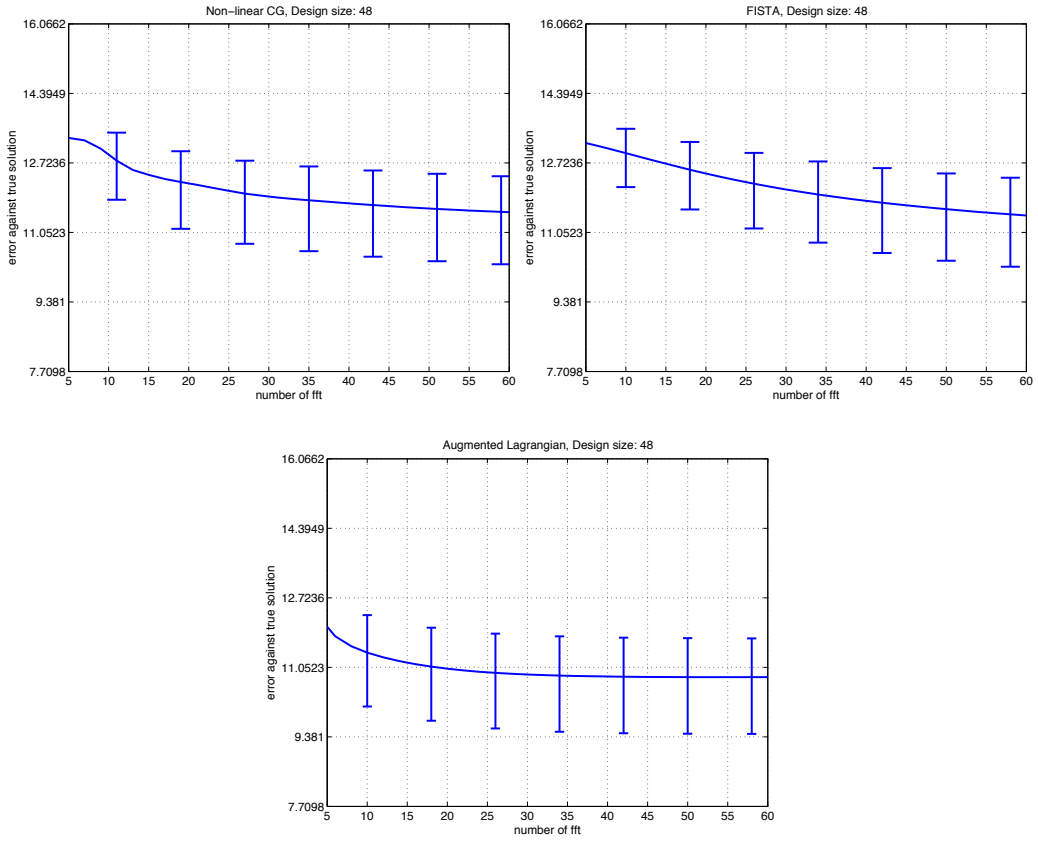



Figure H.13: Design size: 48. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices, plots with error bars).

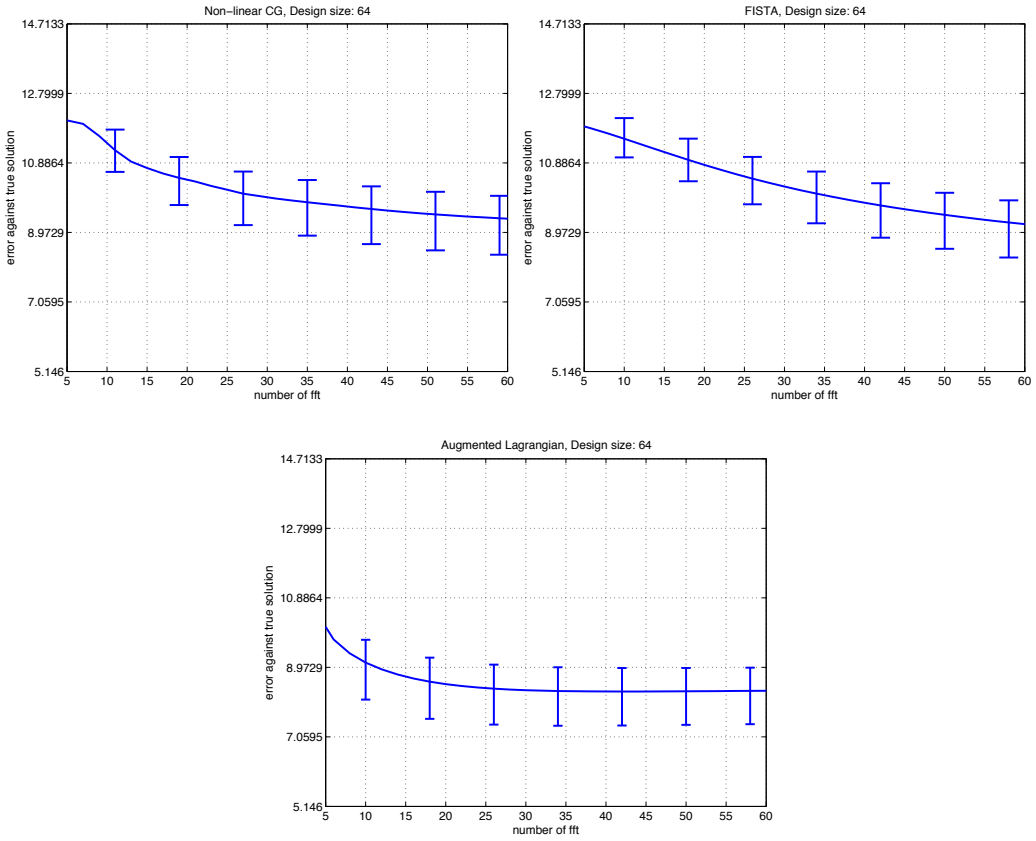



Figure H.14: Design size: 64. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices, plots with error bars).

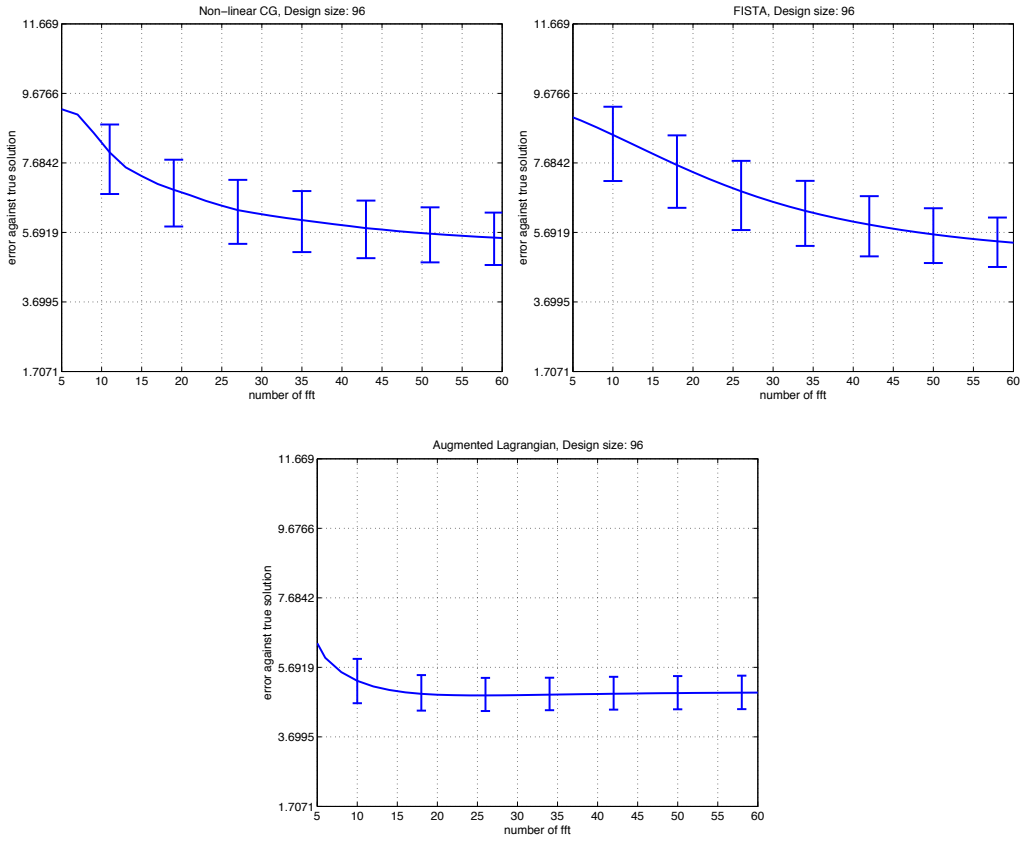



Figure H.15: Design size: 96. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices, plots with error bars).

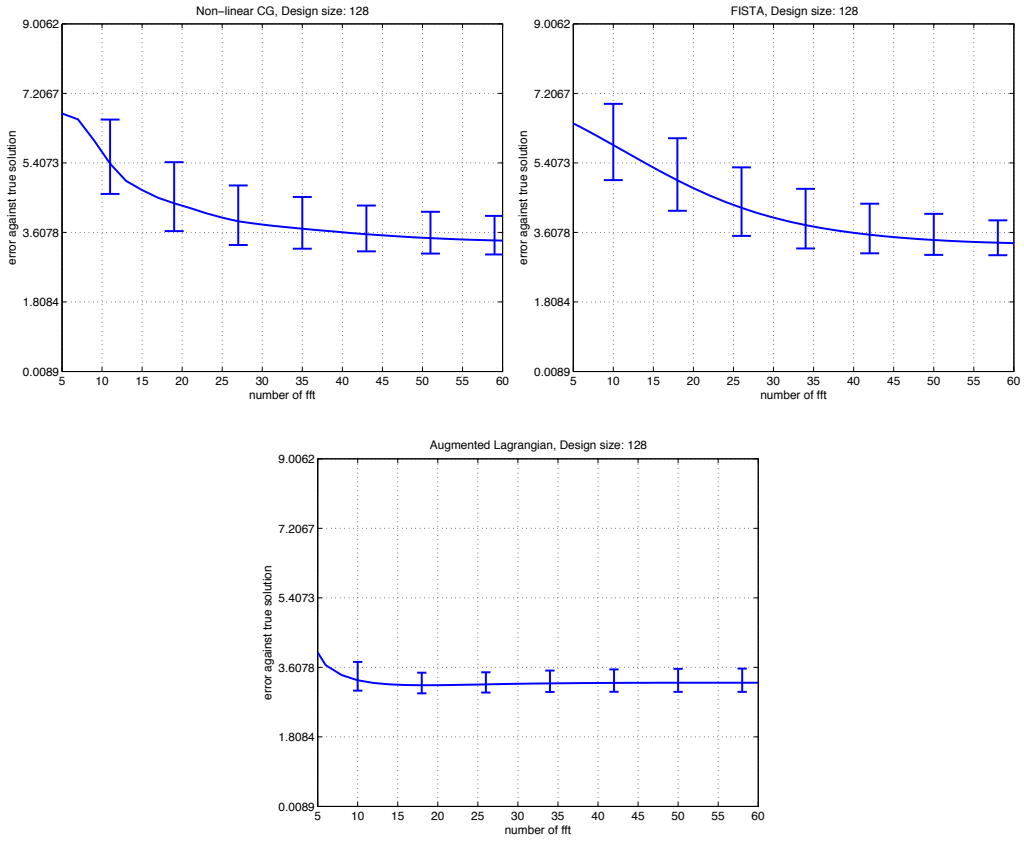



Figure H.16: Design size: 128. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices, plots with error bars).

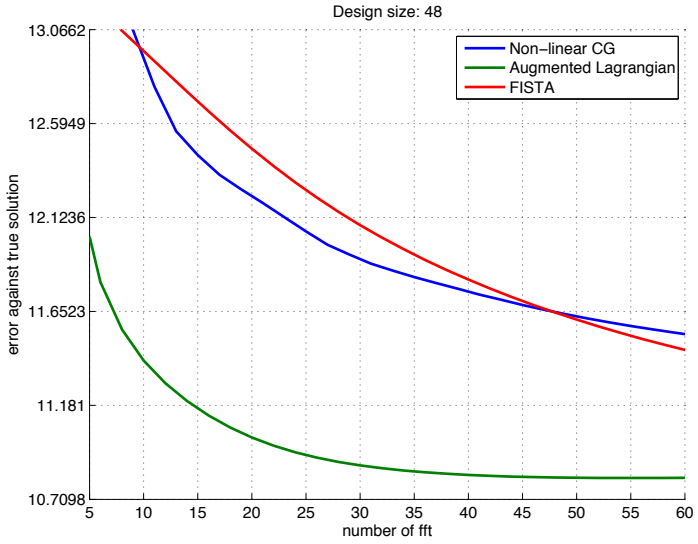



Figure H.17: Design size: 48. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices).

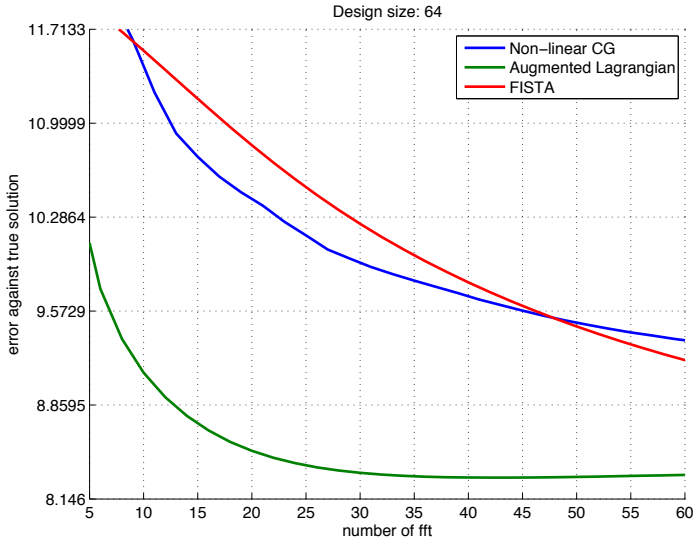



Figure H.18: Design size: 64. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices).

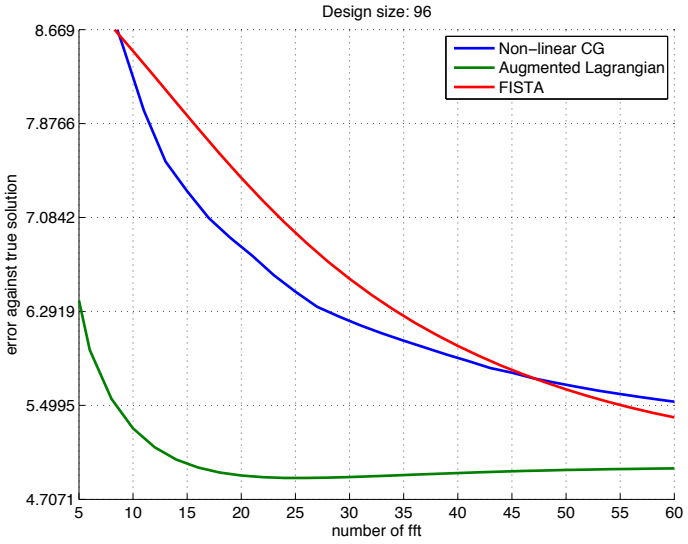



Figure H.19: Design size: 96. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices).

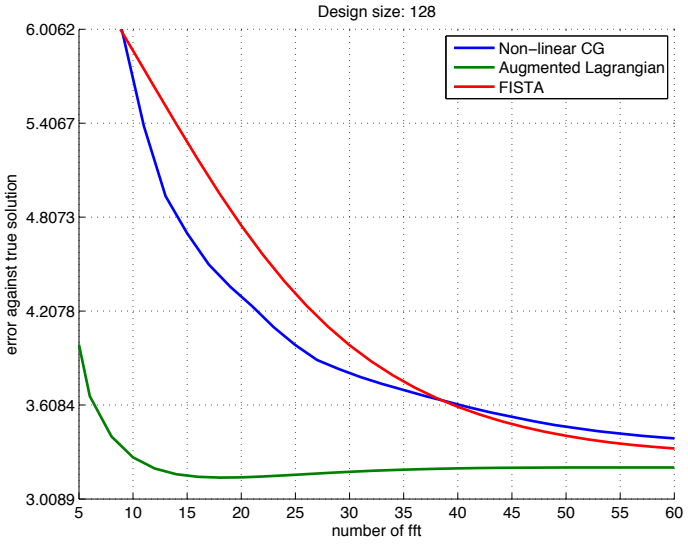
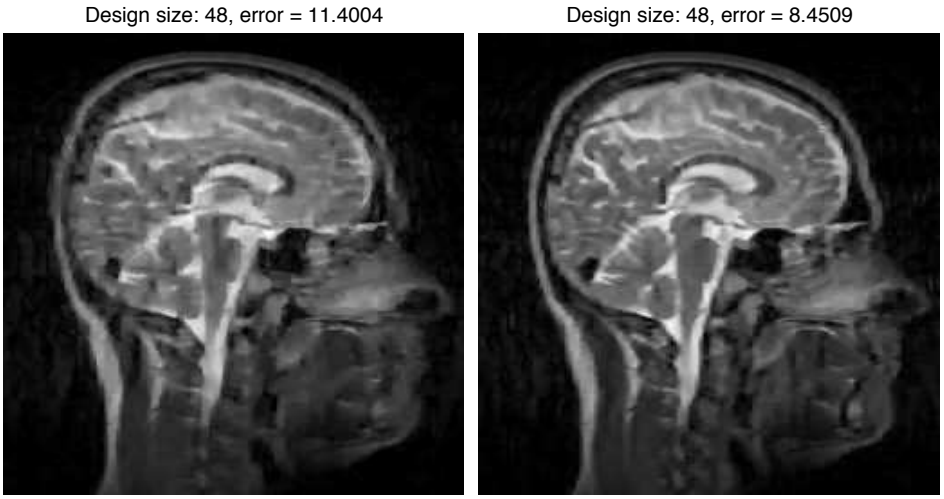
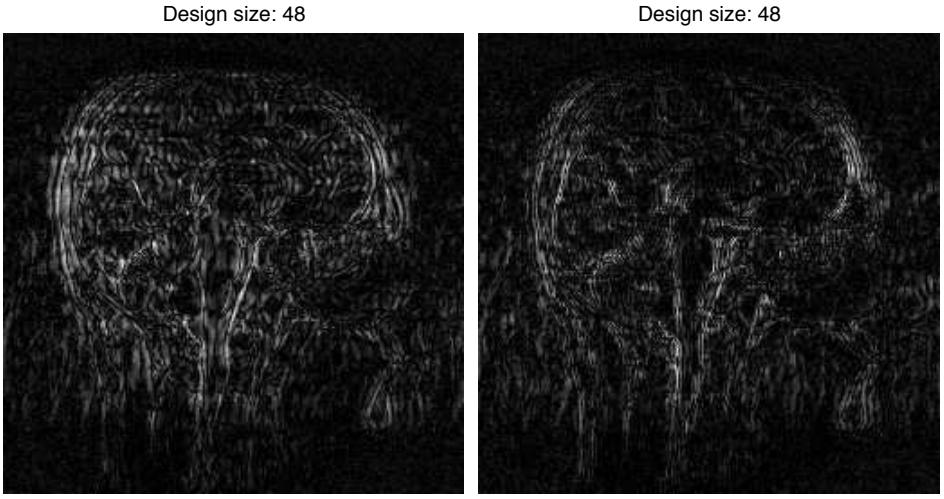



Figure H.20: Design size: 128. The Variable density phase encoding was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices, four different subjects and ten measurement matrices).



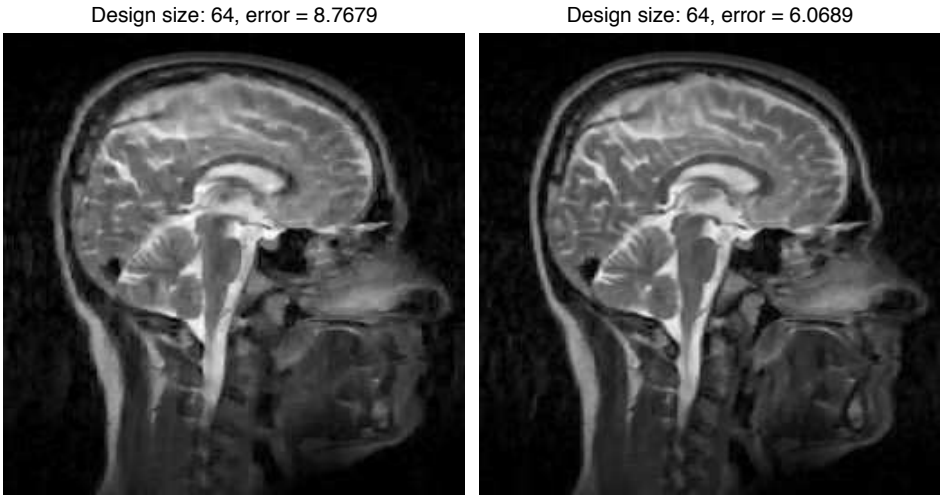

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.21: Design size: 48. The median image for the Variable density phase encoding is shown in Figure H.21a and the reconstructed image for the Bayesian experimental design is shown in Figure H.21b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.



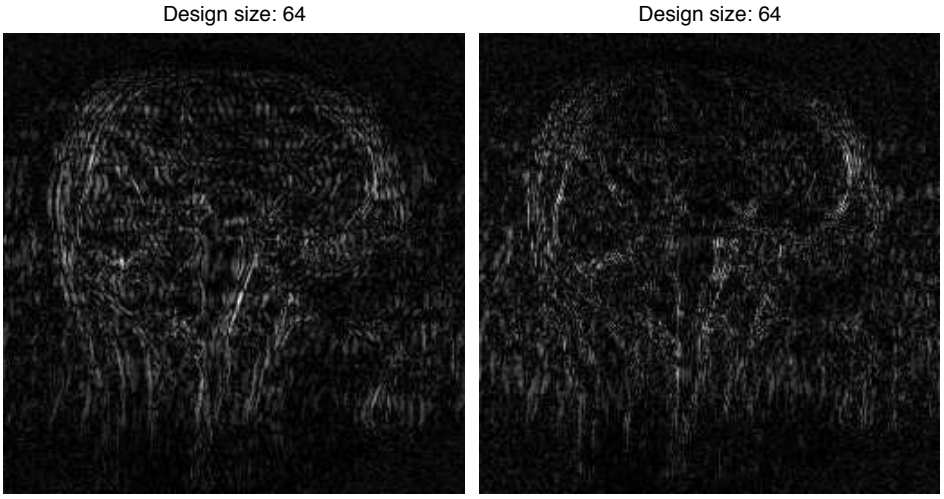

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.22: Design size: 48. Figure H.22a shows the absolute difference between the median image obtained by the Variable density phase encoding and true image. Figure H.22b shows the absolute difference between the reconstructed image obtained by the Bayesian experimental design and true image. Image description: TE=92ms, subject 2, slice 8, sagittal image.



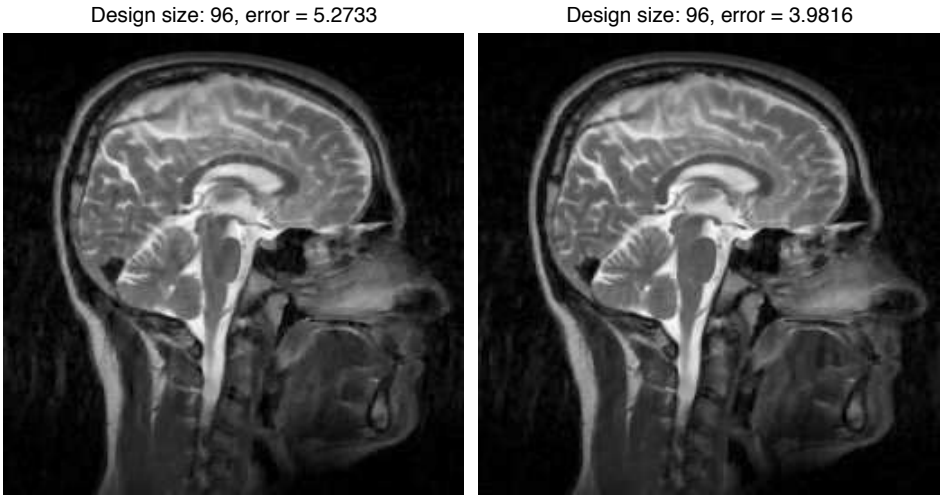

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.23: Design size: 64. The median image for the Variable density phase encoding is shown in Figure H.23a and the reconstructed image for the Bayesian experimental design is shown in Figure H.23b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.



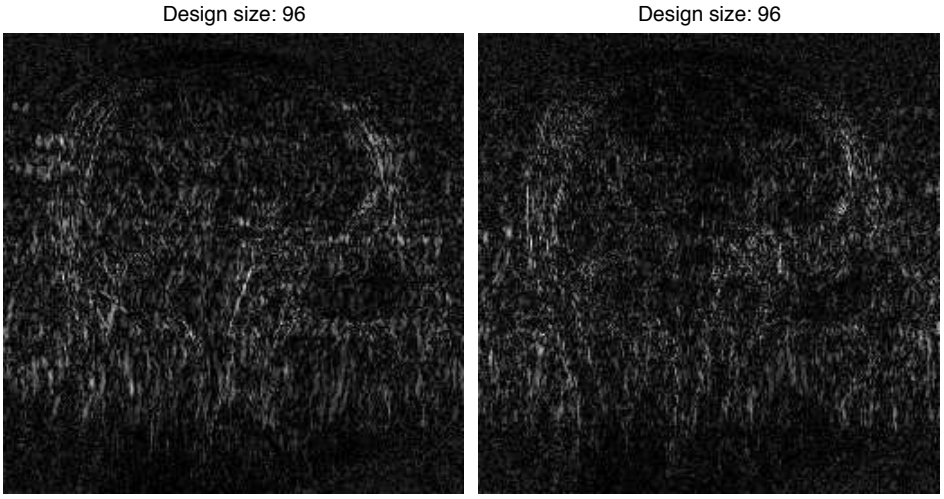

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.24: Design size: 64. Figure H.24a shows the absolute difference between the median image obtained by the Variable density phase encoding and true image. Figure H.24b shows the absolute difference between the reconstructed image obtained by the Bayesian experimental design and true image. Image description: TE=92ms, subject 2, slice 8, sagittal image.



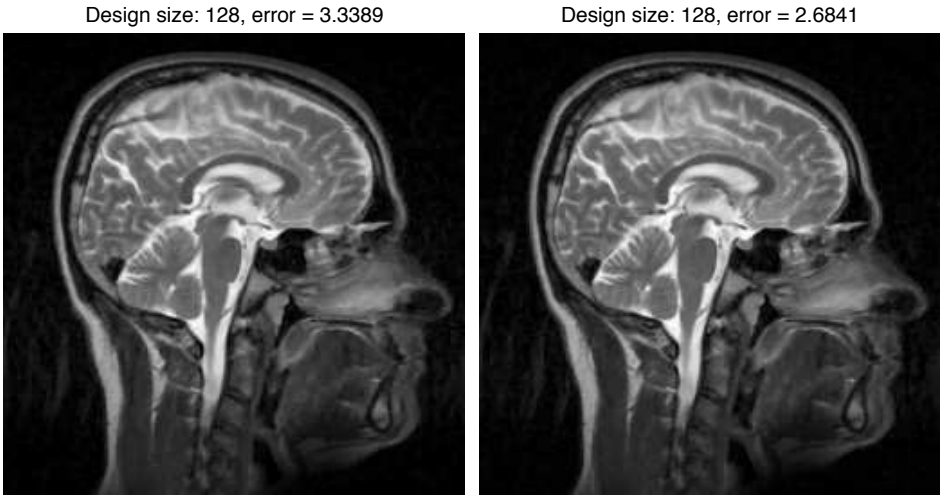

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.25: Design size: 96. The median image for the Variable density phase encoding is shown in Figure H.25a and the reconstructed image for the Bayesian experimental design is shown in Figure H.25b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.



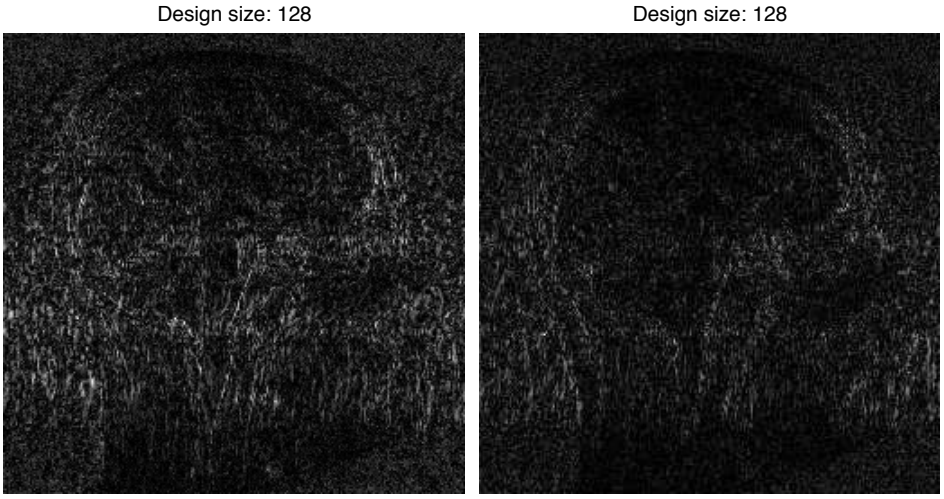

(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.26: Design size: 96. Figure H.26a shows the absolute difference between the median image obtained by the Variable density phase encoding and true image. Figure H.26b shows the absolute difference between the reconstructed image obtained by the Bayesian experimental design and true image. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.27: Design size: 128. The median image for the Variable density phase encoding is shown in Figure H.27a and the reconstructed image for the Bayesian experimental design is shown in Figure H.27b. The number of iterations was limited to 10 steps when the Bayesian experimental design was used, and to 30 steps when the Variable density phase encoding was used. Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Variable Density Phase Encoding (b) Bayesian Experimental Design

Figure H.28: Design size: 128. Figure H.28a shows the absolute difference between the median image obtained by the Variable density phase encoding and true image. Figure H.28b shows the absolute difference between the reconstructed image obtained by the Bayesian experimental design and true image. Image description: TE=92ms, subject 2, slice 8, sagittal image.

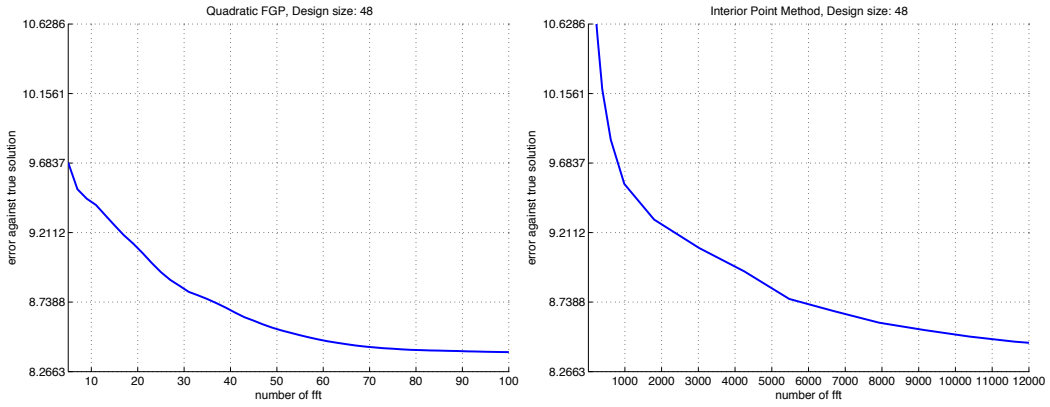



Figure H.29: Design size: 48. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects). The elastic model (7.2) was used.

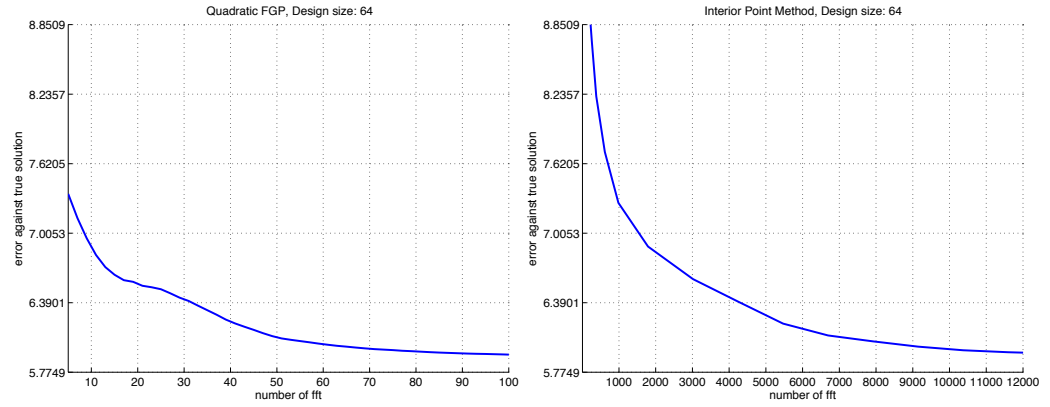



Figure H.30: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects). The elastic model (7.2) was used.

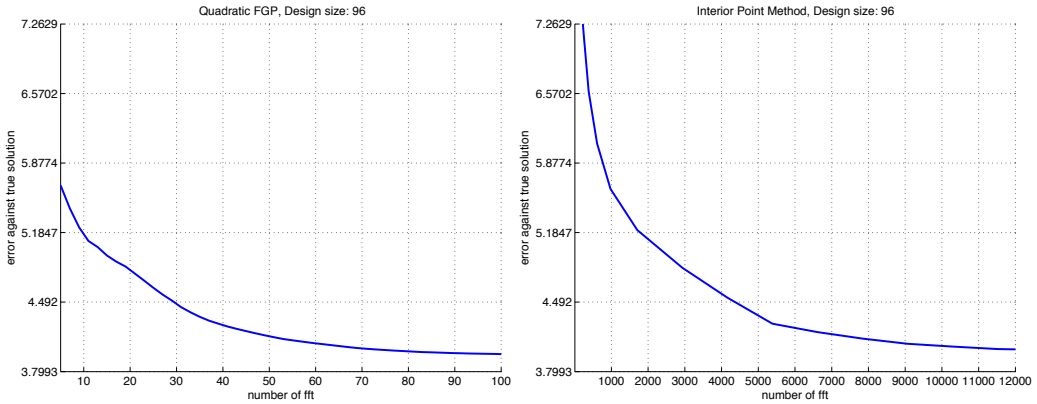



Figure H.31: Design size: 96. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects). The elastic model (7.2) was used.

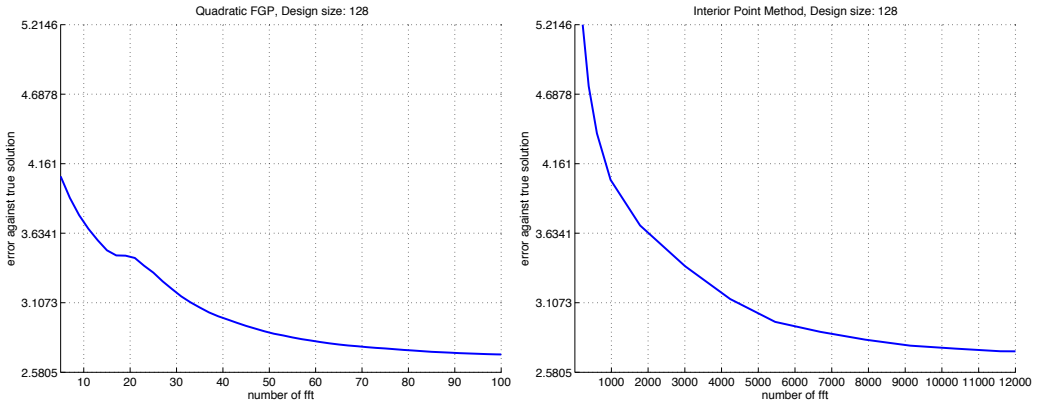



Figure H.32: Design size: 128. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$ (averaged over two slices and four different subjects). The elastic model (7.2) was used.

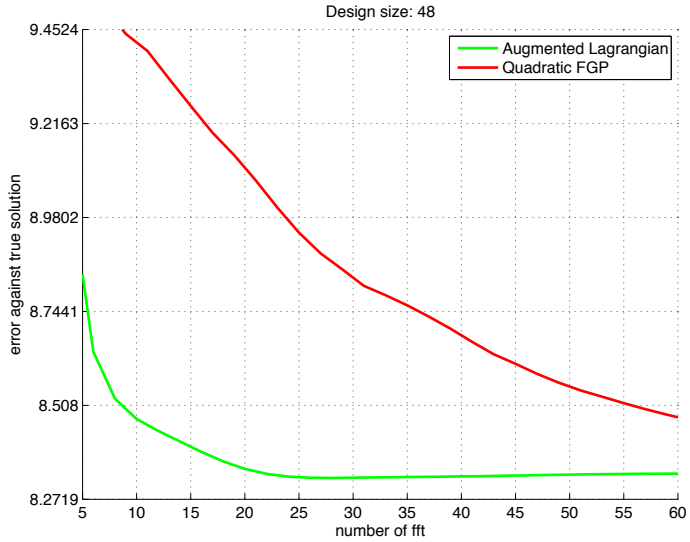



Figure H.33: Design size: 48. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. *Augmented Lagrangian* was used to solve problem (3.7). *Quadratic FGP* was used to solve problem (7.2).

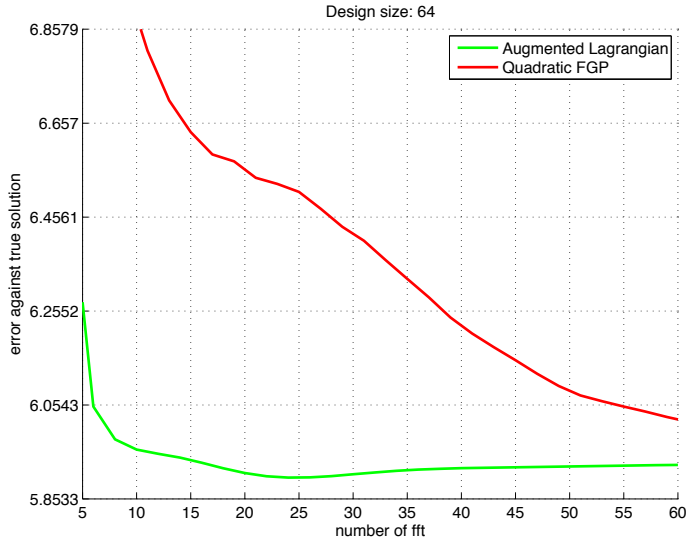



Figure H.34: Design size: 64. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. *Augmented Lagrangian* was used to solve problem (3.7). *Quadratic FGP* was used to solve problem (7.2).

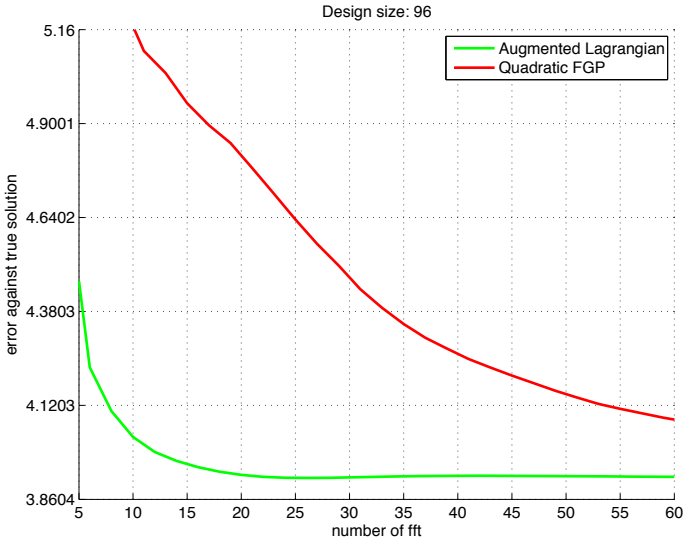



Figure H.35: Design size: 96. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. *Augmented Lagrangian* was used to solve problem (3.7). *Quadratic FGP* was used to solve problem (7.2).

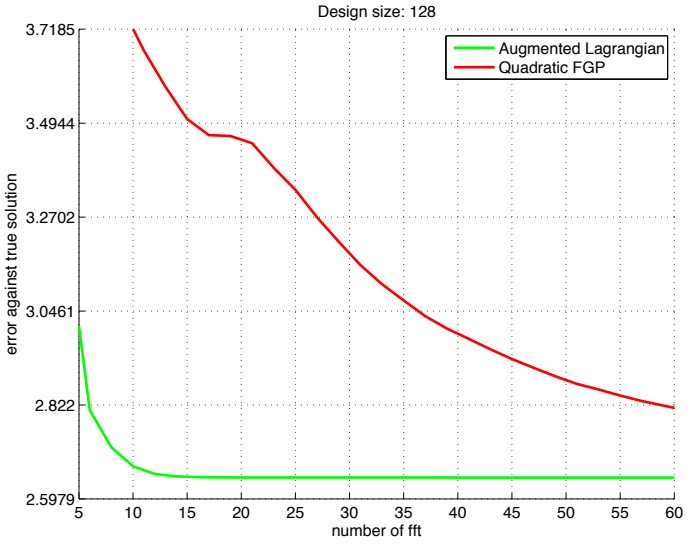
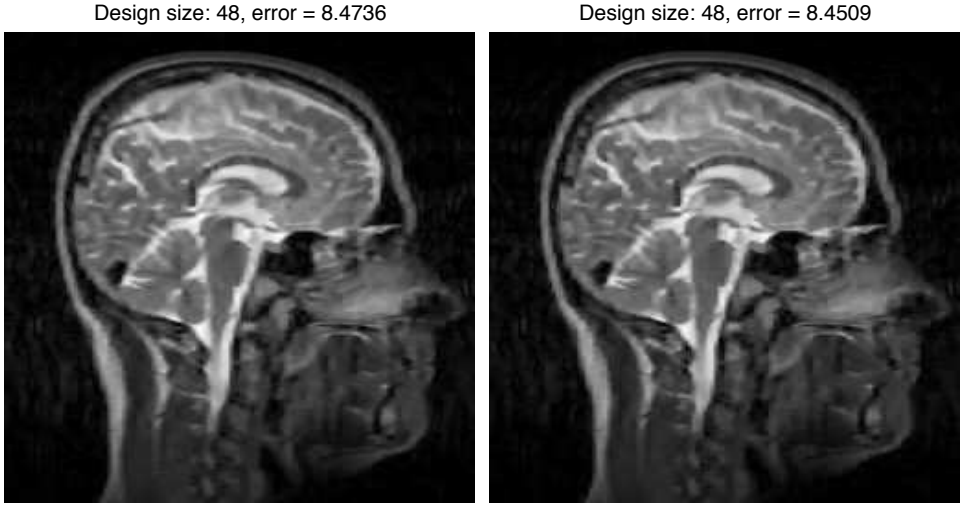
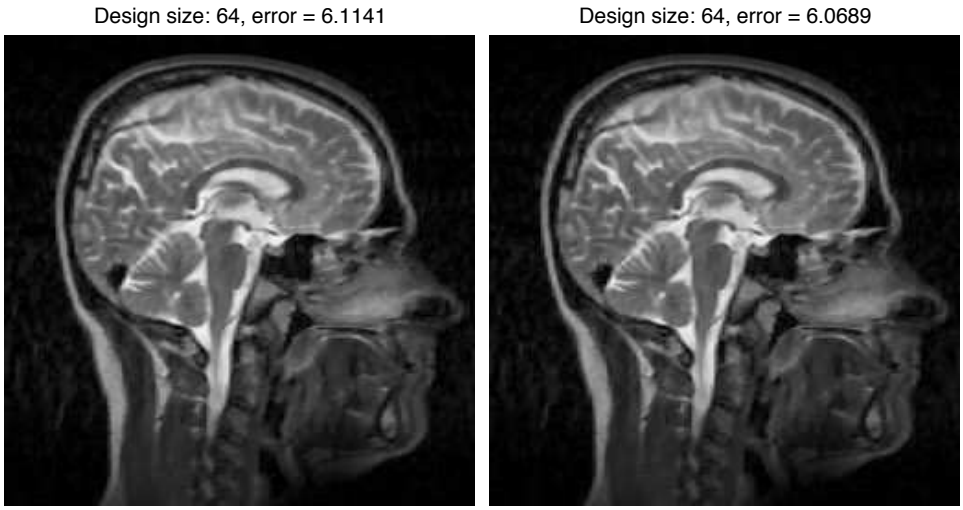



Figure H.36: Design size: 128. The Bayesian experimental design was used. Shown are $l_2$ distances to $\boldsymbol{u}_{true}$. *Augmented Lagrangian* was used to solve problem (3.7). *Quadratic FGP* was used to solve problem (7.2).



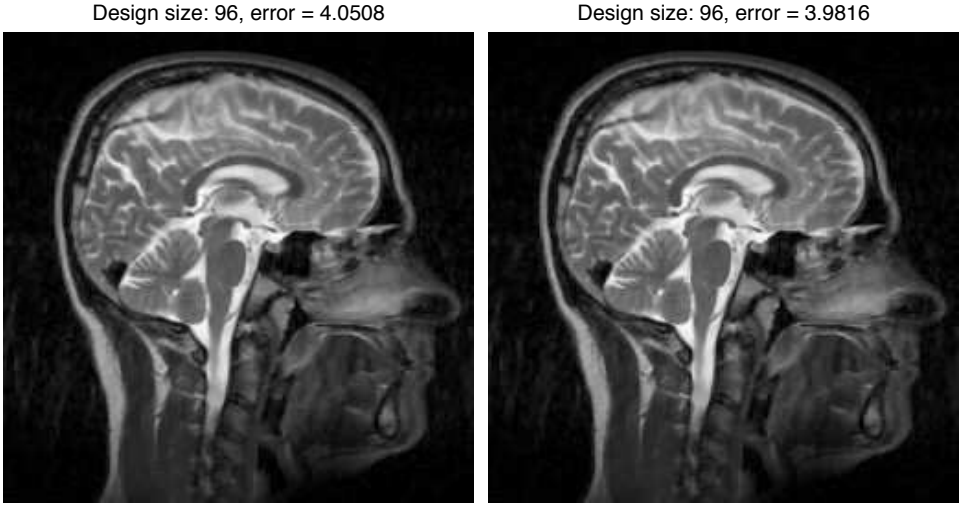

(a) Elastic extension (b) Original model

Figure H.37: Design size: 48. The Bayesian experimental design was used. Figure H.37a shows the reconstruction made by *Quadratic FGP* with model (7.2). Figure H.37b shows the reconstruction made by *Augmented Lagrangian* with model (3.7). Image description: TE=92ms, subject 2, slice 8, sagittal image.



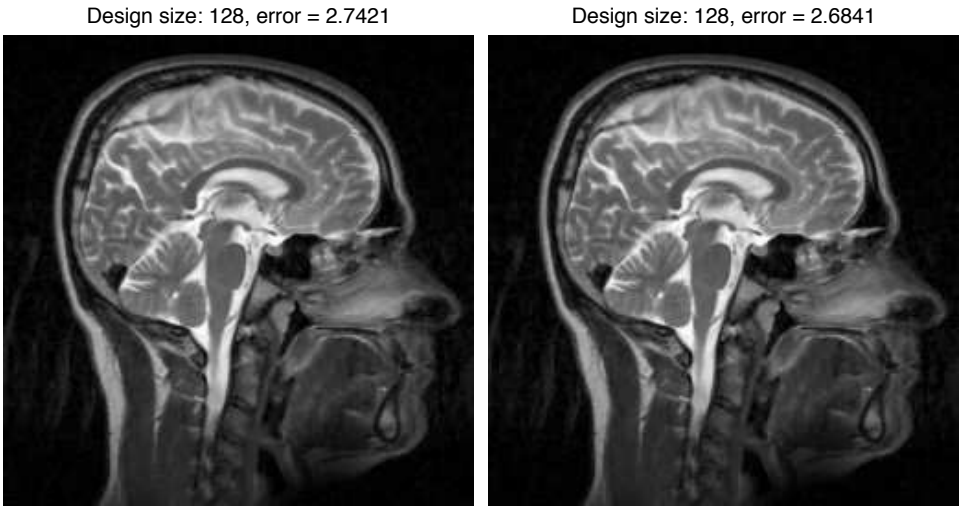

(a) Elastic extension (b) Original model

Figure H.38: Design size: 64. The Bayesian experimental design was used. Figure H.38a shows the reconstruction made by *Quadratic FGP* with model (7.2). Figure H.38b shows the reconstruction made by *Augmented Lagrangian* with model (3.7). Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Elastic extension (b) Original model

Figure H.39: Design size: 96. The Bayesian experimental design was used. Figure H.39a shows the reconstruction made by *Quadratic FGP* with model (7.2). Figure H.39b shows the reconstruction made by *Augmented Lagrangian* with model (3.7). Image description: TE=92ms, subject 2, slice 8, sagittal image.




(a) Elastic extension (b) Original model

Figure H.40: Design size: 128. The Bayesian experimental design was used. Figure H.40a shows the reconstruction made by *Quadratic FGP* with model (7.2). Figure H.40b shows the reconstruction made by *Augmented Lagrangian* with model (3.7). Image description: TE=92ms, subject 2, slice 8, sagittal image.