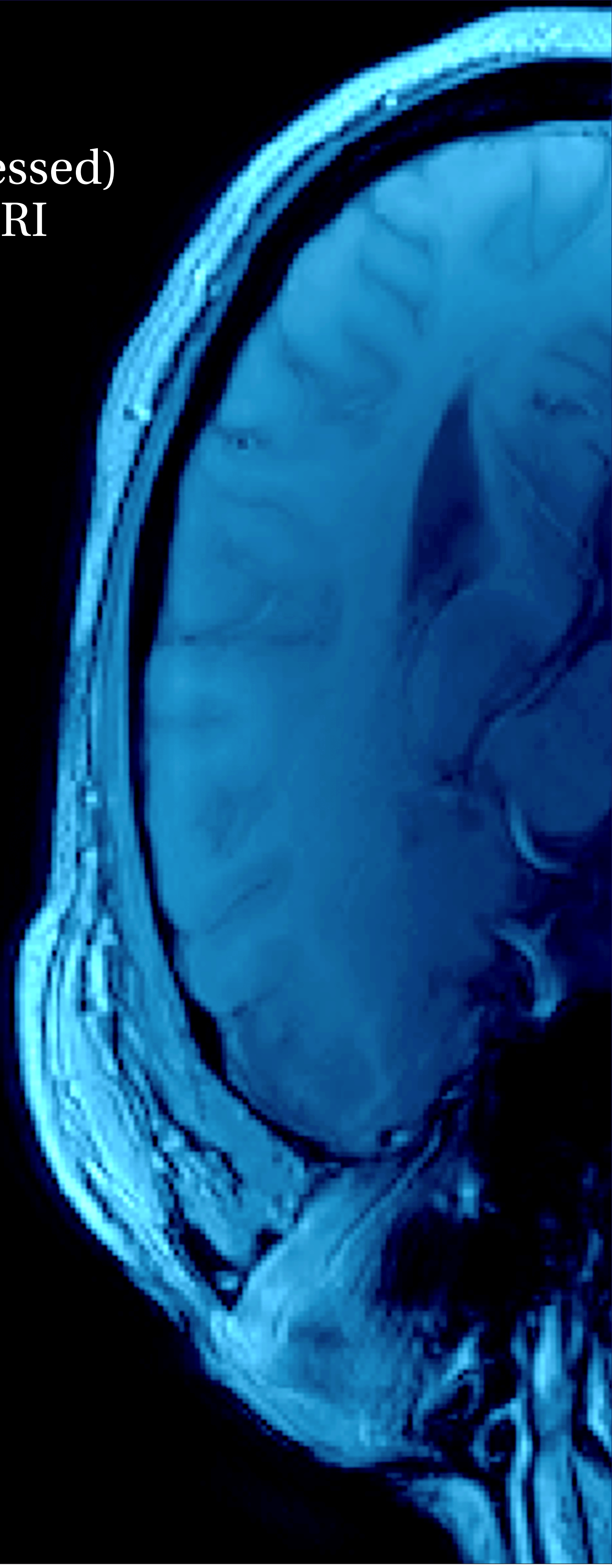
Accelerating (Compressed) SENSE Scans in MRI



Kriti Dhingra

Delft University of Technology

Challenge the future

Accelerating (Compressed) SENSE Scans in MRI

by

Kriti Dhingra

in partial fulfillment of the requirements for the degree of

Master of Science
in Electrical Engineering

at the Delft University of Technology,
to be defended publicly on Monday October 26, 2020 at 14:00 hour.

Student Number:	4897757	
Supervisor:	Dr. ir. R. F. Remis	
Thesis committee:	Dr. N. V. Budko,	TU Delft
	Dr. ir. J. H. F. van Gemert,	Philips
	ir. A. Moerland,	Philips

Abstract

Magnetic Resonance Imaging is a painless procedure to produce high-resolution diagnostic images. Today, it is one of the essential clinical imaging modalities. One of the major challenges involved with this imaging modality is its long scanning time. Parallel imaging in combination with compressed sensing has overcome this challenge to a great extent. As a quid pro quo for this reduced scan time is the increase in image reconstruction time. An extensive research is focused to develop algorithms to make the image reconstruction faster. Fast Iterative Shrinkage Threshold algorithm is one of these algorithms (which is clinically viable) that speeds up the image reconstruction process.

The present project focuses to speed up the particular algorithm, fast iterative shrinkage threshold algorithm, by preconditioning the convex optimization problem. This work proposes two new preconditioners, specifically in the context of the given algorithm, but otherwise can be used with different frameworks solving similar problems. The first preconditioner is a degree one polynomial of the system matrix and the second preconditioner is a block diagonal matrix where each block is a circulant matrix. The preconditioners are evaluated using two stopping criteria: residual error and relative error. The computation complexity of both the preconditioners are evaluated by measuring the floating point operations and total time consumption. Additionally, the simulations are performed by undersampling the data at two factors $r = 2$ and $r = 4$.

The results indicate that the polynomial preconditioner reduces the overall time by a factor of 0.25 however is computationally expensive to construct. On the other side, block diagonal circulant preconditioner is extremely cheap to construct and evaluate on a vector but does not provide the desired results within the current framework. The study concludes that a suitable preconditioner for FISTA is the one that without affecting the largest eigenvalue of the system matrix improves the condition number and simultaneously is cheap to construct and evaluate.

Acknowledgement

After almost a year (the most bizarre one because of COVID-19) of this research work, I am finally entitled to receive the reward for which I sweat my blood for the past two years. My journey was no less than a roller coaster ride, however, this experience was the most wonderful one. I am delighted to express my gratitude to every one who has been a part of this journey, no matter how short or long their role was.

I express my immense gratitude to my academic supervisor Dr. ir. Rob Remis for providing me with the privilege to carry out the research work under his guidance. I am thankful for his constant support, cooperation and feedback during the course of this project. Thank you for introducing me to this world of MRI!

I thank Jeroen from the bottom of my heart because I can not even imagine finishing this project without his support. I really respect him for his broad knowledge and his enthusiasm to continuously learn new things. He has provided me his valuable time and indispensable advices at every aspect of this project. I am truly glad and thankful to have a mentor like him.

I would also extend my gratitude to Ad Moerland for providing me an opportunity to work at Philips and for arranging all the administrative work. I am also thankful to him for checking up on me especially during the uncertain lock-down. I express my sincere regards to Dr. Neil Budko for his time and interest to be a part of my thesis committee.

My masters degree would not have been possible without the support and encouragement of my family members. I am grateful to have parents who made me believe in the saying that ’the sky’s the limit’ and I find myself very fortunate to be born in such a family.

This acknowledgement will be incomplete without thanking my friends who constantly encouraged and supported me. Ishita, I am indebted to her for things she has done for me. From the brainstorming sessions to helping me write my report to making me meals and to make sure that I am okay, she has done it like my mother would do it for me. I am grateful to have such a caring and supportive friend. Mudit, you are a great friend and an amazing chef. Thank you for being there and hearing me out, every time. Further, I am thankful to Rishabh who without even his presence helped me in the best possible way. You guys are no less than my own family.

Undoubtedly, it was a stressful year and I would not have managed without the support of Prernna, Anil, Kanya and my research fellows. Even though it was stressful, we definitely made some fun memories.

Kriti Dhingra
Delft, 26 October 2020

Contents

List of Figures

List of Tables

Acronyms

BCCB	Block Circulant with Circulant Block. 13
CS	Compressed Sensing. 1
FISTA	Fast Iterative Shrinkage Threshold Algorithm. 2
FLOPS	Floating Point Operations. 38
ISTA	Iterative Shrinkage Threshold Algorithm. 9
MRI	Magnetic Resonance Imaging. 1
PI	Parallel Imaging. 1
SENSE	SENSitivity Encoding. 5

1 Introduction

Introduction of medical imaging techniques in 1895 brought a whole new perspective in the field of diagnosis with the discovery of non-invasive diagnostic techniques. Amongst them, a significant and widely used technique is Magnetic Resonance Imaging (MRI). History of MRI can be traced back to the 1970's and its foundations even back to the 1930's. MRI is based on the phenomenon of Nuclear Magnetic Resonance (NMR) where atomic nuclei possessing certain magnetic properties when subjected to a strong static magnetic field are perturbed by the selective absorption of radiofrequency (RF) waves. This phenomenon was discovered and validated by Isidor Isaac Raabi in 1938 [1]. Later, in the 1970's, Paul C. Lauterbur used this phenomenon to create MR images [2]. Since then, significant developments have taken place producing commercial MR scanners which in turn has made MRI a clinically viable imaging modality.

1.1. Motivation

Today, MRI is an essential non-invasive technology for radiologists. Its importance can be owed to three factors: (i) no risk of radiation exposure (ii) tomographic scanning and (iii) high-resolution anatomical images. It uses the NMR phenomenon to generate high-contrast images to look inside the human body. However, despite being a vital and pervasive imaging modality, it has a major drawback of long scanning time. This poses a problem for claustrophobic patients who have to remain inside the enclosed machine for long scan sessions. Additionally, this reduces the patient's compliance which degrades the quality of the scan. The latter problem occurs especially with pediatric patients. These difficulties stimulated the research to find methods that accelerate the scan session time in MRI.

The invention of Parallel Imaging (PI) in the late 1900's addressed this well-motivated problem. PI uses multiple independent receiver coils, which are spatially sensitive, to measure the NMR signal. The spatial sensitive coils help in localising the NMR signal themselves, thereby, reducing the phase-encoding steps and accelerating the scan session. In other words, this means that the measured data is lesser than that suggested by the Nyquist criterion leading to aliasing artefacts in the image. Still, it is possible to obtain the true image. This is achieved by using the information of the sensitivity profiles of each coil in solving the reconstruction problem. However, there is a theoretical limit up to which a scan session can be accelerated. To accelerate it further, Compressed Sensing (CS) methods are used with PI. CS allows measuring even less data than suggested by the Nyquist criterion and recovers the image by using prior information. This prior information exploits the sparsity representation of the image in a certain transformed domain. Combination of PI and CS is used to obtain high-accelerated scan sessions.

A complete scan session includes the data acquisition time and the time taken to reconstruct the image. Although PI and CS reduce the data acquisition time they subsequently increases the image reconstruction time. The increased reconstruction time can be owed to the additional work that has to be performed because of data undersampling which results in an ill-posed model of huge size. Solving such problems with direct methods is infeasible. Hence, approaches involving iterative procedure are explored to obtain the solution of these problems. The current model of PI-CS is formulated as a basis pursuit problem where the objective is to find the minimum ℓ_1 norm solution to an underdetermined linear system. Using conventional gradient-based algorithms for basis pursuit has a major drawback of slow convergence rate, thus, restraining their use. Alternatively methods that solve the ℓ_1 norm problems are considered. The literature has abundant ℓ_1 solvers with its particular application in medical imaging. Selection of a particular algorithm depends on the user implementation as some may provide a stable solution at the cost of slow performance whereas others might prove to be faster methods with unstable behaviour. For clinical purposes, stable methods are of course the reliable choice.

To further improve the clinical application it is required that the method is not only stable but also exhibits faster convergence. One way to achieve this is by preconditioning the system with a matrix that in some sense approximates the inverse of the system matrix. This improves the condition number of the linear system, thereby, improving the rate of convergence. However, finding a preconditioner is a challenge, especially when the system matrix is only available as a callback function that performs the matrix-vector product. Implicit system matrix constraints the found preconditioner to be available as a callback function. This stems the need to find suitable preconditioning techniques for clinically viable iterative solvers such that the overall performance of the MR system is improved.

1.2. RESEARCH QUESTIONS

Preconditioning techniques in general context is a well-researched area, however its application in PI-CS problems ([3][4][5][6]) is still in its evolving stage. The research considers the Fast Iterative Shrinkage Threshold Algorithm (FISTA) as the ℓ_1 solver and aims to find a suitable preconditioner to accelerate the image reconstruction time for parallel imaging-compressed sensing model with Cartesian trajectories. To realize the objective, the study intends to find answers to the following relevant research questions.

- Is it feasible to fit preconditioner in the FISTA framework without compromising the quality of the reconstructed image?
- Is it possible to find a preconditioner that increases the convergence without affecting the true solution?
- Due to memory limitations, it is not possible to explicitly store the preconditioner matrix. Is it possible to implement the preconditioner as an operator?
- Can the preconditioner be extended to different frameworks?

1.3. THESIS OUTLINE

The report is structured as follows: chapter 2 provides a brief introduction to MRI followed by parallel imaging in MRI. It also formulates the parallel imaging problem model and introduces the concept of compressed sensing with parallel imaging. Chapter 3 describes FISTA and the existing preconditioning techniques. The implementation of preconditioned FISTA with existing preconditioners is described in chapter 4. Further, chapter 5 provides the derivation of the two new preconditioners and their computational implementation is given in chapter 6. Chapter 7 presents the results and its discussion. Lastly, chapter 8 concludes the results and describes the limitations and future work of the study.

2

Parallel Imaging and Compressed Sensing

The chapter begins with a short introduction to MRI and its working which is followed by a detailed discussion on parallel imaging in MRI and the techniques to reconstruct the image in that case. Further, it describes the principle of compressed sensing and how it can be combined with parallel imaging to reduce the reconstruction time.

2.1. Magnetic Resonance Imaging

To be able to comprehend the necessity of PI and its development, a brief description of the working of MRI is presented. As mentioned before, MRI is based on the principle of the NMR phenomenon and consequently requires nuclei that are NMR active. Generally, nuclei with an odd mass number or odd atomic number which have a spin property attached to them are known to be NMR active. In our body, one such nucleus which is available in abundance is, Hydrogen atom (^{1}H). The spinning of these hydrogen nuclei induces a magnetic moment making them act like a small magnet.

MRI utilizes a strong magnetic field that is in the order of 1.5 to 7T. This strong and static magnetic field exerts a rotary force (or torque) on the magnetic moment of the nuclei which aligns the nuclear spins of hydrogen atoms with that of the field. This system is then said to be in equilibrium. Now, when a radio frequency current is exposed to this system, the system gets excited by the electromagnetic energy of the RF wave thereby, making the system go out of equilibrium. Once this RF field is turned off, the spin system returns to equilibrium by emitting radio-waves which carries information of different types of tissues. These RF waves are proportional to the number of hydrogen atoms, the magnitude of the external magnetic field and the time at which the waves are measured after the system is excited with an RF pulse. Thus, to generate the MR image, the signals are spatially localised by the use of gradient coils that emit a semi-static strong magnetic field to spatially encode the frequencies. It is important to note that the NMR phenomenon occurs at the resonant frequency which in MRI is defined as Larmor frequency given by the Eq. 2.1.

$$\omega = \gamma B \tag{2.1}$$

where ω is the resonant frequency, γ is the gyromagnetic ratio (for nucleus of ^{1}H = 42.58 MHz T^{-1}) and B is the external magnetic field applied to the nuclear spin system. Gradient coils yield the Larmor frequency of the system as a function of position. This means different positions of the system are excited with different frequencies to encode the spatial position. This is called the frequency encoding step which can be used to (i) select a slice in two-dimensional imaging and (ii) to localise the signal within a slice. However, frequency encoding can localise signal only in one dimension within the slice i.e. if it is applied in the horizontal direction then it is not possible to discriminate between the pixels of a particular column. For reference, a representation of the same is shown in Fig. 2.1. Although signals from each column can be separated because each column is excited with different frequency but it is not possible to separate signals from various pixels corresponding to the same column. For example, the red and green pixels in Fig. 2.1 can not be separated if the signals are localised in one dimension. To be able to localise in the second dimension, spatial encoding is done by a method known as phase encoding. This is also performed by the gradient coils in MR but this

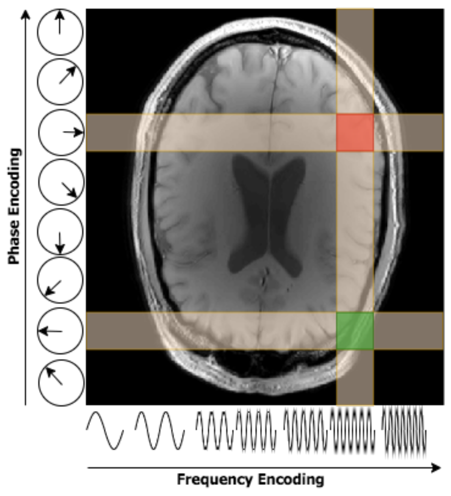


Figure 2.1: Representation of frequency and phase encoding in a slice.

gradient is used for a different purpose. In this case, gradient coils excite the system for a very short time in which every position experiences a (short) different rotational speed yielding position dependant phases which helps to localise signal in the vertical direction corresponding to a particular frequency. Intuitively, it can be seen as a coordinate system where the x-direction has varying frequencies whereas y-direction has varying phases of the signal (shown in Fig. 2.1). It is important to realise that the measured signal is spatially sampled in the frequency domain, commonly referred to as k-space. Therefore, the true image is obtained by taking the Fourier transform of the measured k-space.

To obtain the image, the signal must be sampled with at least the number of samples that satisfies the Nyquist-Shanon sampling theorem, implying that the phase encoding steps must be repeated several times. The repetition of the encoding steps to produce a high-resolution image consumes most of the time in MRI. To get an idea, an MRI scan may take about 20 to 60 minutes depending upon the type and amount of slices taken during the scan. This stems the need for research to develop methods which reduce the time consumption of MRI, leading to the invention of PI.

2.2. PARALLEL IMAGING

Initially, the MRI set up used only a single receiver coil that covered the entire body part which was being scanned resulting in a large field of view (FOV), but at the same time consumed a lot of time for data acquisition. On the contrary, PI uses multiple spatial sensitive coils where each coil works as an independent receiver such that all the coils can operate simultaneously to reduce the data acquisition time. The concept behind PI is easily understood with the simplified version of 'parallel imaging with localized sensitivity' (PILS) method [7].

Consider imaging the axial plane of the brain with a single receiver coil. Assume that it requires N_y phase encoding steps to sample the FOV. It implies that the time required to image the plane would be proportional to N_y. Now, instead of a single coil consider imaging the same axial plane with two receiver coils on either side of the brain as shown in Fig. 2.2. In this case, the upper(lower) coil will be sensitive to the signals from the upper(lower) half of the plane. So, instead of sampling the full FOV, only half of the FOV is sampled (in k-space), resulting in an aliased image which is sensitive to upper(lower) half of the brain as depicted in Fig. 2.2(a) (2.2(b)). Sampling half the FOV will require only half of the Ny phase encoding steps and hence in half the acquisition time. The aliased images from both of the coils can now be successfully combined to reconstruct the un-aliased brain as shown in Fig. 2.2(c).

The above example explains the concept of PI in a facile form but in practice, the procedure involves some complications. The key problem associated with PI is that while sampling half the FOV (in k-space), the Nyquist-Shanon sampling theorem is violated which results in aliasing artefacts in the image, as shown in fig 2.3a and 2.3b. To obtain an un-aliased (Fig. 2.3c), PI requires a reconstruction algorithm that combines the signals from all the coils and removes the aliasing artefacts.

Before understanding the methods to reconstruct images in PI, it is worthwhile to highlight two important points related to the hardware setup of PI. First, the multiple receiver coils used in PI must be arranged such

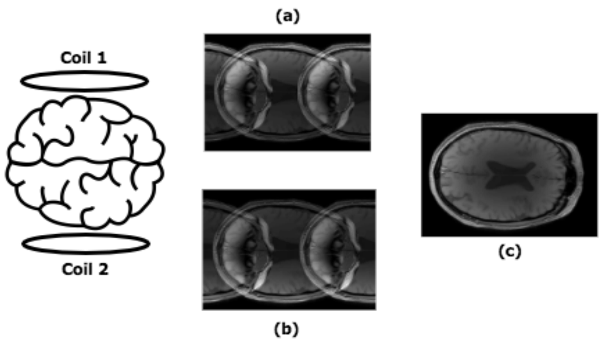


Figure 2.2: Simplified concept of parallel imaging.

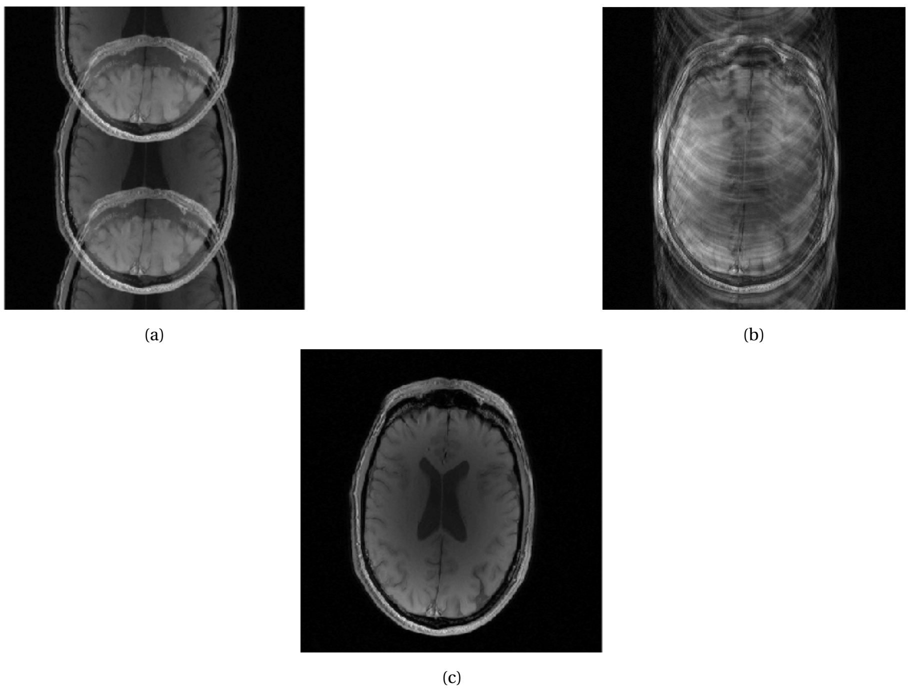


Figure 2.3: Reconstructed images for undersampled and fullysampled FOV in PI. (a) Aliased artefacts due to uniform undersampling. (b) Aliased artefacts due to random undersampling. (c) Un-aliased image.

that every coil has a different spatial sensitivity pattern over the FOV. Second, it is important to have the knowledge of coil's sensitivity profiles which is used in the reconstruction algorithm to remove the aliasing artefacts. Intuitively, this can be seen as providing some degree of information related to spatial encoding which is why it is possible to reduce phase encoding steps in PI.

Owing to the sensitivity profiles, the desired image can be constructed with an algorithm that combines data from all the coils. In PI, the reconstruction algorithms can be grouped into two categories based on the domains they operate on i.e. image domain or k-space. Although, several methods, irrespective of the operated domain, namely SENSE [8], GRAPPA[9], SMASH[10], SPIRiT[11] and ESPIRiT[12] have been derived for Cartesian trajectories, yet only SENSE and GRAPPA are clinically viable. SENSE operates in the image domain whereas GRAPPA operates in k-space. For the present study, we restrict ourselves to SENSE technique to reconstruct the image.

2.2.1. PARALLEL IMAGING RECONSTRUCTION: SENSE

SENSitivity Encoding (SENSE) is a parallel imaging reconstruction method that operates in the image domain indicating that the aliasing artefacts caused due to the undersampling of k-space are rectified in the image domain. It is achieved by weighting each pixel of the image by the respective pixel's spatial sensitivity. As mentioned before, the desired image and k-space are related by the Fourier transformation, which is mathematically formulated as

$$\mathrm{F}\mathbf{x} = \mathbf{m}_{\text{full}} \tag{2.2}$$

where, $\mathbf{m}_{\text{full}} \in \mathbb{C}^{N\times 1}$ is the fully sampled measured k-space, $\mathbf{x} \in \mathbb{C}^{N\times 1}$ is the true image and $\mathrm{F} \in \mathbb{C}^{N\times N}$ is the two-dimensional discrete Fourier transformation matrix. Here, the 2-D true image has the dimensions $m \times n$ and $N = mn$. Similarly, the relation in PI can be formulated by weighting the true image in Eq. 2.2 with the coil sensitivity profiles. Considering that the k-space is fully sampled, including the additive noise, the relation can then be modified as

$$\mathrm{F}\mathrm{S}_i\mathbf{x} = \mathbf{m}_{\text{full},i} \quad \text{for } i = 1,\ldots,\mathrm{N}_c \tag{2.3}$$

where, $\mathbf{m}_{\text{full},i} \in \mathbb{C}^{N\times 1}$ are the fully sampled measured k-space with noise for $i \in \{1,\ldots,\mathrm{N}_c\}$ with N_c denoting the number of receiver coils and $\mathrm{S}_i \in \mathbb{C}^{N\times N}$ are diagonal matrices containing the coil sensitivity patterns of each coil. In practice, the k-space data in PI is undersampled to accelerate the data acquisition time, which slightly modifies the Eq. 2.3 by including the undersampling pattern in the problem formulation. The modified formulation is represented as the following.

$$\mathrm{R}\mathrm{F}\mathrm{S}_i\mathbf{x} = \mathbf{m}_i \quad \text{for } i = 1,\ldots,\mathrm{N}_c \tag{2.4}$$

where, $\mathbf{m}_i \in \mathbb{C}^{N\times 1}$ are the undersampled k-space data (including noise) for $i \in \{1,\ldots,\mathrm{N}_c\}$ with zeros filled at the non-measured points and $\mathrm{R} \in \mathbb{R}^{N\times N}$ is a diagonal matrix representing the undersampling pattern. For 2-D cartesian trajectories, the freedom for undersampling is only in the direction of phase-encoding direction. In such a case, there are two potential sampling patterns i.e. uniform and random undersampling patterns. Uniform undersampling measures every r^{th} line in the phase-encoding direction whereas random undersampling measures $1/r^{th}$ lines in the phase encoding direction that are chosen randomly, here r is the acceleration factor. An example of these patterns for $r = 2$ is shown in Fig. 2.4. It is observed that uniform

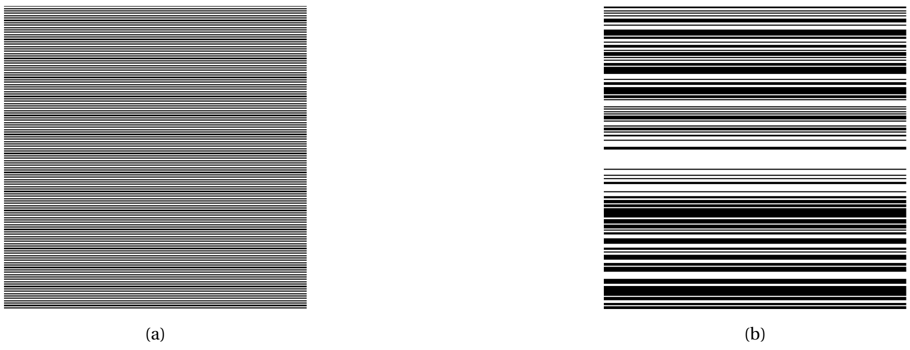


Figure 2.4: Undersampling patterns for 2-D cartesian trajectory. (a)Uniform undersampling pattern. (b)Random undersampling pattern.

undersampling pattern, Fig. 2.4a, results in coherent aliasing artefacts as shown in Fig. 2.3a whereas random undersampling pattern, Fig. 2.4b, results in incoherent artefacts as depicted in Fig. 2.3b. These incoherent artefacts due to random undersampling makes this pattern the desirable choice when used in compressed sensing. This is discussed in detail in section 2.3.

Eq. 2.4 indicates that for each coil, the number of unknowns i.e. the components of vector $\mathbf{x}$ are more than the number of equations. Therefore, the formulated problem for each coil is an ill-posed problem. But it is possible to reconstruct an estimate of the true image $\mathbf{x}$ by combining the measurements of each coil to solve the following least square problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathrm{N_C}} \|\mathrm{R}\mathrm{F}\mathrm{S}_i\mathbf{x} - \mathbf{m}_i\|_2^2 \right\} \tag{2.5}$$

where $\hat{\mathbf{x}} \in \mathbb{C}_{N\times 1}$ is the estimated image. Theoretically, the estimated image, $\hat{\mathbf{x}}$ will closely match the true image, $\mathbf{x}$ until $r \leq N_c$, which makes sense because for $r > N_c$ the least squares problem 2.5 itself becomes an ill-posed problem. However, in practice the limit is never reached due to noisy measurements. To accelerate the data acquisition time beyond the number of coils, compressed sensing techniques can be combined with PI, detailed discussion of this can be found in the following sections.

2.3. COMPRESSED SENSING

The method of Compressed Sensing (CS) was first introduced as a mathematical concept in information theory. It suggested the method to recover a signal when it is considerably undersampled with respect to the Nyquist-Shanon sampling theorem [13][14]. It can be regarded as relying on the concept of compression, which focuses on finding a sparse representation of data or signals. CS provides an edge over the other compression techniques because of low computational cost and complexity while acquiring the data. CS is of specific interest to MRI because it enables to measure significantly less k-space data while maintaining the resolution of the true image.

In order to implement CS for MRI applications, there are two requirements:

1. Sparse Representation: The desired image must have a sparse representation in a known transform domain.

2. Incoherent Artefacts: Artefacts caused due to undersampling of k-space should be incoherent (behaves like noise) in the transformed domain.

These requirements can be easily satisfied in MRI which is the reason behind successful applications of CS in MRI. Examining the first condition regarding the sparsity constraint, it is observed that MR images, in general, are not sparse, however, they are known to be sparse in a certain transformed domain. Operators that map the vector containing image data to a sparse vector are called sparsifying transforms. These are extensively used for image compression with significant applications of compression strategy in JPEG, JPEG-2000, and MPEG [15]. Some examples of these sparsifying transforms are finite differences, discrete cosine transform (DCT) and discrete wavelet transform (DWT). For MRI, total variation (TV) and wavelet transforms are the popular approaches used by researchers to represent MRI in a sparser domain.

To fulfil the second condition, which intends to use the undersampling pattern that yields incoherent interference, recall the introduction of potential undersampling patterns in section 2.2.1. It was mentioned that randomly undersampling the phase-encoding lines resulted in incoherent artefacts. Therefore, employing random undersampling pattern for data acquisition enables the application of CS in MRI because then MRI meets both the requirements of compressed sensing. To understand the significance of incoherent artefacts for the feasibility of CS, Lustig et al. [16] provided an intuitive example in 1-D which also explains how the signals can be recovered in CS. A similar example is described further.

Consider a 1-D sparse signal as depicted in Fig. 2.5(a). Undersample this signal, uniformly 2.5(b) and randomly 2.5(c), in the frequency domain by a factor of 4. In the uniform undersampling pattern, equispaced points are measured whereas in the random undersampling pattern, randomly chosen points are measured. During this process, the unmeasured values are filled with zeros. Inverting this signal back to temporal domain results in aliasing artefacts, 2.5(d) and 2.5(e) that behaves differently for uniform and random patterns. In the former case, the artefacts are coherent and the resulting signal is superimposed with shifted signal copies which prevent the recovery of true signal 2.5(f). Whereas, in the latter case, the artefacts are incoherent and behave like noise which can be removed by thresholding 2.5(g). Fig. 2.6 depicts the equivalence between random undersampling artefacts and artefacts due to additive noise, where 2.6(a) utilize the same sparse signal as in 2.5. Their equivalence can also be interpreted visually by comparing 2.6(c) and 2.5(e) representing similar artefacts even when both of them used different approaches, wherein the former case noise is added to the sparse signal and in the latter scenario, the sparse signal is undersampled in the frequency domain.
Since, thresholding is the heuristic choice when dealing with additive noise, CS uses the same principle to recover the undersampled signal. It utilizes a nonlinear reconstruction method that iteratively thresholds the signal and selects the prominent components. Subsequently, it also subtracts the interference caused by these components to recover the smaller components. This example provides an elementary background of the compressed sensing principle to comprehend its application in MRI.

2.3.1. COMPRESSED SENSING IN PI

Application of CS in MRI was first proposed by [16] where Lustig et al. reconstructed the image by a nonlinear method that promotes sparsity while maintaining data fidelity. This is mathematically modeled as a convex optimization problem of the following form

$$\begin{aligned} &\min |\Psi \mathbf{x}|_1 \\ &\text{s.t. } \|\mathrm{RF}\mathbf{x}-\mathbf{m}\|_2^2 < \epsilon \end{aligned} \tag{2.6}$$

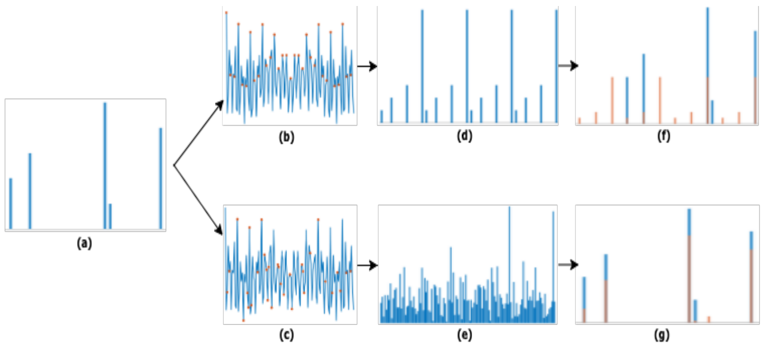


Figure 2.5: A one-dimensional example of compressed sensing. (a) Representation of a sparse signal. (b) Uniformly undersampled frequency domain of sparse signal. (c) Randomly undersampled frequency domain of sparse signal. (d) Coherent aliasing artefacts in temporal domain in case of uniform undersampling. (e) Incoherent aliasing artefacts in temporal domain in case of random undersampling. (f) Ambiguity in recovered signal from uniform undersampling. (g) Recovered signal (orange) from random undersamping.

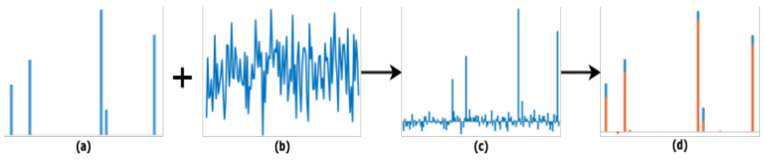


Figure 2.6: A one-dimensional example describing the removal of noise by soft thresholding (ℓ_1 denoising). (a) Representation of a sparse signal. (b) Gaussian noise. (c) Sparse signal with additive gaussian noise. (d) Recovered signal (orange) using ℓ_1 denoising.

where, $|\cdot|$ denotes the ℓ_1 norm, Ψ denotes the sparsifying transform operator and ϵ controls the fidelity of reconstruction to the measured data. Eq. 2.6 can be solved using numerical solvers as described in chapter 3. Combining the compressed sensing approach with PI results in an equation similar to 2.6 which is written as an unconstrained convex optimization problem as described in Eq. 2.7.

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathrm{N_c}} \|\mathrm{RFS}_i\mathbf{x} - \mathbf{m}_i\|_2^2 + \lambda|\Psi\mathbf{x}|_1 \right\} \tag{2.7}$$

where, λ is the regularization parameter for the ℓ_1 norm term. Presence of the non-smooth function, $|\Psi\mathbf{x}|_1$, makes the problem 2.7 difficult to solve as the gradient for the given non-smooth function is not defined at $\Psi\mathbf{x} = 0$. In convex optimization theory, this particular kind of problem where the cost function is the sum of smooth and non-smooth function, as shown below

$$\min_x f(x) + g(x),$$

where, f and g are convex functions and g is non-differentiable, is termed as proximal minimization problem [17]. Several iterative schemes exist in literature to solve these problems and some other that are specifically designed to solve the compressed sensing problems. Discussion about the existing literature on these algorithms and detailed discussion about the method employed in this project for reconstruction is done in the following chapter, 3.

3
Numerical Solvers & Preconditioning Techniques

The chapter mentions the iterative solvers required to solve the minimization problem 2.7. It discusses in detail the Iterative Shrinkage Threshold Algorithm (ISTA) and fast iterative shrinkage threshold algorithm (FISTA). Further, it describes the preconditioning techniques which is followed by a detailed explanation of the ones implemented in this project.

3.1. Numerical Solvers

The relation between the unknown image and fully sampled k-space measurements is formulated as a linear system (Eq. 2.2, 2.3) of the form

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{C}^{N\times N}, \quad \mathbf{x} \in \mathbb{C}^{N}, \quad \mathbf{b} \in \mathbb{C}^{N}. \tag{3.1}$$

If A is nonsingular, such system has a unique solution given by the least square approach: $\mathbf{x} = A^{-1}\mathbf{b}$. In more general methods, A is usually ill-conditioned and it is not possible to find a solution with direct inversion. Additionally, in practice, the k-space lines are undersampled in PI which reduces the rank of A making it an underdetermined linear system with an infinite number of solutions. For this case, the least square estimate is obtained by solving the Eq. 2.5, under the assumption that $r \le N_c$ and that the noise is uncorrelated. This is given by the following equation

$$\hat{\mathbf{x}} = (A^H A)^{-1} A^H \mathbf{b} = A^{\dagger}\mathbf{b} \tag{3.2}$$

where, $A^{\dagger} = (A^H A)^{-1} A^H$ is the Moore-Penrose inverse of A [18]. But particularly in PI, the noise is not uncorrelated with measured data. So, to minimize the noise level in the reconstructed image, noise covariance weighted least square problem is solved which yields the solution

$$\hat{\mathbf{x}} = \left(A^H \Psi^{-1} A\right)^{-1} A^H \Psi^{-1}\mathbf{b} \tag{3.3}$$

where, Ψ denotes the noise covariance matrix. In the case where the measured data is noise normalised or is uncorrelated with the noise, the noise covariance matrix can be replaced by an identity matrix. Computing the solution of linear system by Eq. 3.2 and 3.3 is straightforward, however, for large dimension problems it is very expensive, both computation and memory-wise. Hence, an efficient way is to solve them by iterative procedures. If the cost function is differentiable, then, the classic gradient methods such as steepest descent, conjugate gradient and preconditioned conjugate gradient [19] can be used, which is true in the particular case of Eq. 2.5. However, the PI compressed sensing problem, Eq. 2.7, is not differentiable because of the presence of non-smooth function (ℓ_1 term). This restrains the use of conventional gradient methods for solving Eq. 2.7. So, methods that specifically solve ℓ_1 norm problems are considered.

A classic example to solve this type of problem 2.7 is the iterative shrinkage threshold algorithm (ISTA) [20]. Many different algorithms exist in literature which either differently considers this method or is a generalisation of this algorithm. For instance, proximal forward-backward method [21] can be interpreted as a generalisation of ISTA in the convex optimisation theory. Other examples that belong to this particular class of iterative shrinkage threshold algorithms are expectation-maximization algorithm [22], majorization-minimization algorithm [23], Bregman iterations [24], alternating direction method of multipliers (ADMM) [25] and more. This class of method is computationally effective to solve the problems involving ℓ_1 term than earlier used traditional descent algorithms which consumed a considerable amount of time to reach the solution. In particular, they utilize a matrix-vector product at each iteration followed by a shrinkage step where the prominent computation effort is the matrix-vector product. Thorough theoretical analysis of these methods guarantees a minimizer for the cost function with a substantial rate of convergence.

Several methods have been proposed to speed up these techniques further. Among them the most popular are two-step ISTA (TWIST) [26], fast iterative shrinkage threshold algorithm [27], gradient method for minimizing composite functions [28] and subspace optimization method [29]. These methods significantly improve the convergence rate of earlier stated methods. Other relevant work for the considered problem can be found in [13], [30], [4],[31], [32]. Amongst these, the algorithm that holds importance for this project is fast iterative shrinkage threshold algorithm (FISTA). Its computational simplicity and a significant rate of convergence makes it a potential choice for solving Eq. 2.7.

To be able to understand FISTA, it is important to learn about the iterative shrinkage algorithm which lays the foundation for FISTA. This is covered in Section 3.1.1, followed by FISTA in the section 3.1.2.

3.1.1. ISTA

Iterative shrinkage threshold algorithm is a popular method to solve linear inverse problems with ℓ_1 norm regularization. Such problems are difficult to solve because the differential of ℓ_1 norm is not properly defined around zero. Consider the following model

$$\min_{\mathbf{x}} \left\{ F(\mathbf{x}) \equiv \|A\mathbf{x}-\mathbf{b}\|_2^2 + \lambda|\mathbf{x}|_1 \right\} \tag{3.4}$$

where, λ is the parameter that controls the trade-off between data consistency and sparsity in $\mathbf{x}$. ISTA solves this problem by deconvoluting the problem in two substeps where the first substep evaluates the ℓ_2 norm term and the second substep solves the ℓ_1 norm term in Eq. 3.4. The update equation for the problem 3.4 is given by

$$\mathbf{x}_{k+1} = \mathcal{T}_{\lambda t}\left(\mathbf{x}_k - 2tA^H(A\mathbf{x}_k - \mathbf{b})\right) \tag{3.5}$$

where, t represents the stepsize with $t \in \left(0, 1/\|A^H A\|\right)$ to ensure convergence of $\mathbf{x}_k$ to a minimizer $\mathbf{x}^*$ of Eq. 3.4 and the notation $\|\cdot\|$ depicts the matrix norm. $\mathcal{T}_\alpha : \mathbb{C}^N \to \mathbb{C}^N$ is the soft thresholding (shrinkage) operator defined as

$$\mathcal{T}_\alpha(\mathbf{x}) = (|\mathbf{x}| - \alpha)_+ \operatorname{sgn}(\mathbf{x}) \tag{3.6}$$

where, $(c)_+$ denotes the operation $\max(c, 0)$, $c \in \mathbb{R}$. Here the sgn function is operated on complex vector so, it is defined as $\operatorname{sgn}(\mathbf{x}) = \mathbf{x}./|\mathbf{x}|$.

The shrinkage operator follows from the subdifferential of ℓ_1 norm. It should be noted that Eq. 3.5 can be split as

$$\mathbf{v} = \mathbf{x}_k - 2tA^H(A\mathbf{x}_k - \mathbf{b}) \tag{3.7a}$$

$$\mathbf{x}_{k+1} = (|\mathbf{v}| - \alpha)_+ \operatorname{sgn}(\mathbf{v}). \tag{3.7b}$$

This way it is evident that ISTA solves two subproblems at each iteration where Eq. 3.7a can be viewed as solving the ℓ_2 norm term and equation 3.7b can be viewed as solving the ℓ_1 norm term. Another way to see this is by deriving ISTA as an extension to the gradient-based methods. For reference, one can find the detailed derivation by Beck and Teboulle [27]. With this approach, the algorithm achieves only a sublinear rate of convergence and is perceived to be a slow method.

Algorithm 1: ISTA Iterations

```
1 initialize $t, \mathbf{x}_0 \in \mathbb{C}^N$, $k = 1$
2 while stopping criteria > tolerance do
3     $\mathbf{v} = \mathbf{x}_{k-1} - 2tA^H(A\mathbf{x}_{k-1} - \mathbf{b})$;
4     $\mathbf{x}_k = \max((|\mathbf{v}| - \alpha), 0)\operatorname{sgn}(\mathbf{v})$;
5     $k = k + 1$;
6 end
```

3.1.2. FISTA

FISTA is an extension to the class of iterative shrinkage threshold algorithms (ISTA) [20] which are specifically used to solve large scale linear inverse problems. FISTA is a faster version of ISTA which besides maintaining the computational simplicity of ISTA improves the global convergence rate. It aims to minimize a general model of the following form

$$F(\mathbf{x}) \equiv f(\mathbf{x}) + g(\mathbf{x}) \tag{3.8}$$

where it is assumed that $g : \mathbb{C}^N \to \mathbb{R}$ is a nonsmooth continuous convex function and $f : \mathbb{C}^N \to \mathbb{R}$ is a continuously differentiable ($C^{1,1}$) convex function. The algorithm first approximates the function $f(\mathbf{x})$ in Eq. 3.8 by a linear function using taylor expansion and then proximally regularizes the linearized differentiable function. Further, at each iteration it selects the next iterate as a particular linear combination of the previous two iterates. Mathematical formulation of the approximate model with linearised function and proximal regularisation at a given point $\mathbf{w}$ is represented as

$$Q_L(\mathbf{x},\mathbf{w}) := f(\mathbf{w}) + \langle \mathbf{x}-\mathbf{w}, \nabla f(\mathbf{w})\rangle + \frac{L}{2}\|\mathbf{x}-\mathbf{w}\|_2^2 + g(\mathbf{x}) \tag{3.9}$$

where, L is the Lipschitz constant of $\nabla f(\mathbf{x})$. The function, Eq. 3.9, has a unique minimizer

$$p_L(\mathbf{w}) = \text{argmin}\left\{ f(\mathbf{w}) + \langle \mathbf{x}-\mathbf{w}, \nabla f(\mathbf{w})\rangle + \frac{L}{2}\|\mathbf{x}-\mathbf{w}\|_2^2 + g(\mathbf{x})\right\}. \tag{3.10}$$

After ignoring the constant terms in $\mathbf{w}$, Eq. 3.10 can be reduced to

$$p_L(\mathbf{w}) = \underset{\mathbf{x}}{\text{argmin}}\left\{ g(\mathbf{x}) + \frac{L}{2}\left\| \mathbf{x} - \left(\mathbf{w} - \frac{1}{L}\nabla f(\mathbf{w})\right)\right\|_2^2\right\}. \tag{3.11}$$

Therefore, the update equations of FISTA are described as

$$\mathbf{x}_k = p_L(\mathbf{y}_k) \tag{3.12a}$$

$$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2} \tag{3.12b}$$

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{3.12c}$$

where the next iterate $\mathbf{y}_k$ is a linear combination of the previous two iterates $\mathbf{x}_k, \mathbf{x}_{k-1}$. Now, if we consider the model in Eq. 3.4 for FISTA, Eq. 3.12a results in the update step of ISTA (Eq. 3.5). So, the only difference between ISTA and FISTA is that FISTA takes an additional step to choose the next iterate more smartly. This algorithm significantly improves the computation complexity from sublinear to subquadratic convergence rate. Theoretical proof of the convergence can be found in [27].

Algorithm 2: FISTA Iterations

```
1 initialize $L$, $t_1 = 1$, $k = 1$ $\mathbf{x}_0 \in \mathbb{C}^N$, $\mathbf{y}_1 = \mathbf{x}_0$
2 while stopping criteria > tolerance do
3     $\mathbf{x}_k = p_L(\mathbf{y}_k)$;
4     $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
5     $\mathbf{y}_{k+1} = \mathbf{x}_k + \left(\frac{t_k-1}{t_{k+1}}\right)(\mathbf{x}_k - \mathbf{x}_{k-1})$;
6     $k = k + 1$;
7 end
```

3.2. PRECONDITIONING TECHNIQUES

Preconditioning, in a layman's term, is a technique that transforms a given problem into a form that is easier to solve. It is generally used with numerical solvers to accelerate convergence, thereby reducing the computation complexity. The most useful application of preconditioners can be found with large dimensional ill-posed linear inverse systems. This motivates the choice to use preconditioners for the given problem of compressed sensing. Before beginning with the literature of preconditioners and its application for the current problem, it is important to learn which properties of the system matrix indicate convergence behaviour and how they are improved by using preconditioning techniques.

Essentially a preconditioner reduces the condition number of a matrix. The condition number for an invertible matrix A is defined as

$$\kappa(A) = \|A\|\|A^{-1}\| \tag{3.13}$$

where, $\kappa(A)$ represents the condition number of matrix A and $\|\cdot\|$ denotes the induced matrix norm. As per the definition, condition number depends on the choice of norm but generally a 2-norm condition number is the common choice. This is represented in the following equation

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \tag{3.14}$$

where, $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are maximum and minimum singular values of matrix A, respectively. Intuitively, it measures the sensitivity of the system i.e. amount of change in the solution given the perturbations in the measured signal. In other words, it helps to determine the stability of the system.

The System is categorized as ill-posed when $\kappa(A)$ is very large. When A is singular $\kappa(A)$ is infinity. Therefore, application of a preconditioner must reduce the condition number. A good preconditioner, M, is the one that significantly improves the condition number: $\kappa(MA) \ll \kappa(A)$. Additionally, from Eq. 3.14 it can be interpreted that a good preconditioner will try to keep the singular values clustered because more clustered the values are, more closer the condition number will be to 1. Along with this, it is important that the construction of the preconditioner and its application on a vector is computationally cheap. Thus, a good preconditioner should have the following characteristics:

1. Preconditioner, M, should approximate the inverse of the given matrix, A, in some sense i.e. $MA \approx I$.
2. Construction of M and its evaluation on a vector should be computationally cheap.
3. M should either be efficient in terms of memory storage or have an operator application.

Once a preconditioner is found that fulfils the above conditions, it can be applied to the matrix in the following ways.

- **Left Preconditioner**: The linear system is reformed as

$$MA\mathbf{x} = M\mathbf{b}. \tag{3.15}$$

- **Right Preconditioner**: The linear system is reformed as

$$AM\mathbf{u} = \mathbf{b}, \quad \mathbf{x} := M\mathbf{u} \tag{3.16}$$

  where the system is solved for unknown $\mathbf{u}$.

- **Two-sided Preconditioner**: In this case M is factored as $M = M_L M_R$ and the linear system is reformed as

$$M_L A M_R \mathbf{u} = M_L \mathbf{b}, \quad \mathbf{x} := M_R \mathbf{u}. \tag{3.17}$$

These preconditioned systems can be solved using the same iterative procedures mentioned in the previous chapter with slight changes to include the preconditioner matrix. Although, it may seem straightforward to precondition systems, in practice finding an appropriate preconditioner is, in general, difficult and requires a lot of technical details. Sometimes, a preconditioner reduces the total number of iterations but is computationally expensive to construct. On the other hand, the preconditioner can be computationally cheap to construct but its application on a vector may increase the computation complexity per iteration, which eventually may reflect in total computation complexity. So, the best preconditioner is a combination of both characteristics i.e. that does not significantly affect the per iteration computation complexity and is cheap to construct. The choice of the preconditioner is specific to the formulated problem and depends on the kind of iterative solver utilized.

The literature has abundant preconditioning techniques available in the context of a linear system of equations. A review of the same can be found in [33] where these techniques are classified based on matrix properties such as symmetric positive/semi-positive definite, symmetric indefinite and non-symmetric matrices with a focus on preconditioners for sparse matrices arising from the discretization of partial differential equations. A detailed discussion of these methods can also be found in [19] which covers some useful preconditioned algorithms along with preconditioning techniques in general. It also describes in detail the techniques for parallel preconditioners. Besides these traditional preconditioning techniques, much research work has been focused on the use of preconditioners specifically for MR reconstruction problem. Early examples include the work of Sutton et al. [3] for single coil MR reconstruction in the presence of field inhomogeneities where they use circulant preconditioners inspired by Clinthorne et al. [34]. Later, Ramani and Fessler [35] also employed a circulant preconditioner, developed by Yagle [36], for the PI-CS image reconstruction. Recently, Muckley et al. [5] discovered the circulant preconditioner for non-cartesian trajectories using FISTA and Koolstra et al. [6] determined the preconditioner for cartesian trajectories in PI-CS approach considering the split Bregman iterations.

In this project, preconditioning techniques for the compressed sensing PI problem with cartesian trajectories are considered. As mentioned before, there are numerous preconditioners available in theory however the detailed description of the ones that were implemented are presented.

3.2.1. JACOBI PRECONDITIONER

To find an appropriate preconditioner, the most common choice is Jacobi preconditioner, where it estimates the inverse of a given matrix by its diagonal elements. For a given matrix A, Jacobi preconditioner, M, is given as

$$M = \text{diag}\{A\}^{-1} \tag{3.18}$$

where, diag$\{\cdot\}$ places the diagonal elements of its argument on the diagonal of a matrix. It should be noted that M is a diagonal matrix of the same dimensions as A.

3.2.2. CIRCULANT PRECONDITIONER

Circulant matrices are special kind of Toeplitz matrix where each row is shifted circularly by one element of the previous row. Linear systems with such matrices have a particular advantage of finding the solution by Fast Fourier transforms (FFT) which requires considerably less arithmetic operations than conventional direct methods. Because of its efficient implementation, preconditioners with circulant structure are a desirable choice, especially for linear systems with hermitian Toeplitz structure. For the current study, we particularly consider the preconditioner derived by Koolstra et al. [6] for compressed SENSE reconstruction problem with Cartesian trajectories. The choice to consider the specific preconditioner of [6] is based on the fact that the current study has similar problem model as considered by them. However, there are two main differences. First, that their work utilize prior information from two sparsifying transforms i.e. total variation (TV) and wavelet transform. Second, that their work considers the Split Bregman framework to solve the reconstruction model. On the contrary side, the current study uses only wavelet transform as sparsifying transform and solves the reconstruction problem with FISTA framework. Even then, it is interesting to investigate if it is possible to extend the application of their work for FISTA. Therefore, this section describes the circulant preconditioner of [6] from the perspective of the current problem model, Eq. 2.7.

The work of [6] assume their system matrix to be a Block Circulant with Circulant Block (BCCB). BCCB matrices have a unique property that they can be diagonalised by two-dimensional Fourier transform. For instance, if C is a BCCB matrix then it can be diagonalised in the following ways

$$D_1 = \mathrm{F}C\mathrm{F}^H \quad \text{or} \quad D_2 = \mathrm{F}^H C\mathrm{F}$$

where the diagonal entries, $\mathbf{d}_1(\mathbf{d}_2)$, of diagonal matrices $D_1(D_2)$ can be efficiently computed as $\mathbf{d}_1 = \mathrm{F}\mathbf{c}_1(\mathbf{d}_2 = \mathrm{F}\mathbf{c}_2)$. Here, $\mathbf{c}_1(\mathbf{c}_2)$ denotes the first row (column) of C. This property is exploited to construct the circulant preconditioner. Therefore, assuming A to be a BCCB matrix, A can be written as the following

$$\mathrm{A} = \mathrm{F}^H \underbrace{\mathrm{FAF}^H}_{K} \mathrm{F}$$

where, K is given by

$$\mathrm{K} = \sum_{i=1}^{N_c} \underbrace{\mathrm{FS}_i^H \mathrm{F}^H}_{C_i^H} \mathrm{R} \underbrace{\mathrm{FS}_i \mathrm{F}^H}_{C_i}. \tag{3.19}$$

Since an assumption is made on A, matrix K is not diagonal. However, K is a diagonal dominant matrix containing the information of $\mathrm{S}_i^H \mathrm{F}^H \mathrm{RFS}_i$. So, by removing the off diagonal elements of K it is still possible to find an approximation of A^{-1}. Thus, by computing the diagonal of K an estimate for the inverse of A is computed which serve as the preconditioner for the given problem. If $\mathbf{k}$ represents the diagonal elements of K then the preconditioner is formulated as

$$M = \mathrm{F}^H \operatorname{diag}\{\mathbf{k}\}^{-1}\mathrm{F}. \tag{3.20}$$

The diagonal elements $\mathbf{k}$ of K are found as

$$\mathbf{k} = \mathrm{F}^H \left\{ \left[\mathrm{F} \sum_{i=1}^{\mathrm{N_c}} \left(\mathbf{c}_{1;i}^H \circ \mathbf{c}_{1;i}^T \right)^T \right] \circ \mathrm{F}\mathbf{r} \right\}. \tag{3.21}$$

where $\mathbf{c}_{1;i}^H$ represents the first row of matrix C_i^H. Thus, the first row $\mathbf{c}_{1;i}^H$ of matrix C_i^H is computed as $\left(\mathbf{c}_{1;i}^H\right)^T = \mathrm{F}^H \mathbf{s}_i^H$ where $\mathbf{s}_i^H$ is the column vector with diagonal entries of matrix S_i. Detailed derivation of Eq. 3.21 can be found in [6].

This preconditioner achieves a speedup factor of 2.5 for the split Bregman framework with computing only two additional FFTs per iteration. The present study investigates the performance this circulant preconditioner with FISTA framework. The implementation and results of the same can be found in the following chapters.

4 Implementation of Preconditioned FISTA

The previous chapters set the groundwork to understand the work of this project where the mathematical model of parallel imaging compressed sensing MR problem is formulated in chapter 2 and the numerical solvers for the problem are discussed in chapter 3. Chapter 3 also explains the use of preconditioning techniques to speed up the iterative solvers, mentions the existing preconditioners and discusses a few of them in detail. Having attained the background, this chapter describes the mathematical implementation of FISTA and preconditioned FISTA.

4.1. Mathematical Model

The study considers the problem of parallel imaging with compressed sensing in MRI, derived in chapter 2 (Eq. 2.7), where Daubechies 4 wavelet transform is used as the sparsifying operator. The problem formulation with the wavelet operator W is described as

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathrm{N_c}} \|\mathrm{RFS}_i\mathbf{x} - \mathbf{m}_i\|_2^2 + \lambda|\mathrm{W}\mathbf{x}|_1 \right\}. \tag{4.1}$$

Although, it is possible to reconstruct the image for the above model (Eq. 4.1), the problem in general is ill-posed. Solving such a system is hard because it is highly sensitive to small perturbations in measurements. One key alternate to make the system better-posed is by providing additional information about the solution. This is achieved by regularising the problem with some prior information. For instance, to better condition the problem 4.1, regularization term, $\|\mathrm{Q}\mathbf{x}\|^2$ can be added where the matrix Q denotes the information that distinguishes the region of interest from the background. With the regularization term, the PI compressed sensing model 4.1 is re-formulated as the following:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathbf{N_c}} \|\mathrm{RFS}_i\mathbf{x} - \mathbf{m}_i\|_2^2 + \lambda|\mathrm{W}\mathbf{x}|_1 + \mu\|\mathrm{Q}\mathbf{x}\|_2^2 \right\} \tag{4.2}$$

where,

$$\mathrm{Q} = \left(\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{S}_i \right)^{-\frac{1}{2}} \tag{4.3}$$

and μ is the regularization parameter that quantifies the reliability of this additional information.

4.1.1. FISTA

To solve the minimization problem 4.1, the project utilizes the fast shrinkage threshold algorithm described in chapter 3, section 3.1.2. Previously, FISTA was described for the general model 3.8 which can now be transformed for the current problem by substituting $f(\mathbf{x}) = \sum_{i=1}^{\mathrm{N_c}} \|\mathrm{RFS}_i\mathbf{x} - \mathbf{m}_i\|_2^2$ and $g(\mathbf{x}) = \lambda|\mathrm{W}\mathbf{x}|_1$. FISTA update equations can then be obtained by solving Eq. 3.11 for the aforementioned function values. After the substitution of $f(\mathbf{x})$ and $g(\mathbf{x})$, the solution is obtained by taking the derivative of Eq. 3.11 as described below

$$\frac{\partial p_L(\mathbf{w})}{\partial \mathbf{x}} = \mathrm{W}^H \lambda \,\mathrm{sgn}\,(\mathrm{W}\mathbf{x}) + \frac{2L}{2}\left[\mathbf{x} - \left(\mathbf{w} - \frac{1}{L}\nabla f(\mathbf{w})\right)\right]. \tag{4.4}$$

Equating the above result (Eq. 4.4) to zero and replacing $\mathbf{w}$ with $\mathbf{x}_{k-1}$ then yields

$$\mathbf{x} + \mathrm{W}^H \frac{\lambda}{L}\,\mathrm{sgn}(\mathrm{W}\mathbf{x}) = \mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1}) \tag{4.5}$$

which when solved for $\mathbf{x}$ gives the following closed form solution

$$\mathbf{x} = \begin{cases} \mathrm{W}^H(\mathrm{W}\mathbf{v} + \frac{\lambda}{L}) & \mathrm{W}\mathbf{v} < \frac{\lambda}{L} \\ 0 & |\mathrm{W}\mathbf{v}| < \frac{\lambda}{L} \\ \mathrm{W}^H(\mathrm{W}\mathbf{v} - \frac{\lambda}{L}) & \mathrm{W}\mathbf{v} > \frac{\lambda}{L} \end{cases} \tag{4.6}$$

where, $\mathbf{v} = \mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1})$. Here, $1/L$ represents the step size where L is the Lipschitz constant of $\nabla f(\mathbf{x}_{k-1})$. Alternatively, Eq. 4.6 can efficiently be written in the following form

$$\mathbf{x}_k = \mathcal{T}_\alpha\left(\mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1})\right) \tag{4.7}$$

where $\alpha = \frac{\lambda}{L}$ and $\mathcal{T}_\alpha$ is the shrinkage operator defined as

$$\mathcal{T}_\alpha(\mathbf{v}) = \mathrm{W}^H\,(|\mathrm{W}\mathbf{v}| - \alpha)_+\,\mathrm{sgn}\,(\mathrm{W}\mathbf{v}). \tag{4.8}$$

Here, the gradient $\nabla f(\mathbf{x})$ is given by

$$\nabla f(\mathbf{x}) = 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i \mathbf{x} - 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathbf{m}_i. \tag{4.9}$$

Eq. 4.9 can equivalently be seen as

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}, \tag{4.10}$$

where, $A = 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i$ and $\mathbf{b} = 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathbf{m}_i$. With the given $\nabla f(\mathbf{x})$, the smallest Lipschitz constant is given by $L = 2\sigma_{\max}(A)$, where $\sigma_{\max}(\cdot)$ denotes the largest singular value of A. FISTA iterations for this model is summarized in algorithm 3. In the above algorithm, $A\mathbf{x}$ is computationally the most expensive step

Algorithm 3: FISTA Iterations for PI-CS Model

```
1 initialize $L$, $\lambda$, $t_1 = 1$, $k = 1$ $\mathbf{x}_0 \in \mathbb{C}^N$, $\mathbf{y}_1 = \mathbf{x}_0$
2 while stopping criteria > tolerance do
3     $\mathbf{v} = \mathbf{x}_{k-1} - \frac{1}{L}(A\mathbf{x}_{k-1} - \mathbf{b})$;
4     $\mathbf{x}_k = \mathrm{W}^H \max((|\mathrm{W}\mathbf{v}| - \alpha), 0)\,\mathrm{sgn}\,(\mathrm{W}\mathbf{v})$;
5     $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
6     $\mathbf{y}_{k+1} = \mathbf{x}_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}_k - \mathbf{x}_{k-1})$;
7     $k = k + 1$;
8 end
```

of FISTA. Although, A remains constant throughout the algorithm but because of large matrix dimensions it is practically impossible to store this matrix. Therefore, $A\mathbf{x}$ can only be evaluated in an operator form. It can be inferred from the gradient Eq. 4.10 that calculation of $\mathbf{b}$ is also a product of matrices and vector, which is again a computationally expensive product. However, the silver lining is that $\mathbf{b}$ is a constant term and need not be calculated for every iteration, thus, saving some complex matrix-vector computations per iteration.

It is important to note that when the absolute value function (ℓ_1 term or in this case $g(\mathbf{x})$) is a function of an operator such as $|\Psi(\mathbf{x})|_1$ where Ψ is unitary sparsifying transform operator (i.e. $\Psi^H\Psi = I$), then the shrinkage operator defined in Eq. 3.6 can be modified as

$$\mathcal{T}_\alpha(\mathbf{x}) = \Psi^H \mathcal{T}_\alpha(\Psi(\mathbf{x})). \tag{4.11}$$

Proof of the same can be found in Appendix B.

FISTA for Regularized Model

Previous section derives the update equations of FISTA for the problem formulation 4.1. In a similar manner, it possible to derive the update steps for regularized problem formulation 4.2. It can be noted that the difference between the two problem formulations lies only in the function value of f, where for regularized problem $f(\mathbf{x}) = \sum_{i=1}^{N_c} \|\mathrm{RFS}_i\mathbf{x} - \mathbf{m}_i\|_2^2 + \mu\|\mathrm{Q}\mathbf{x}\|_2^2$. This changes the gradient value of $f(\mathbf{x})$ for regularized problem to the following

$$\nabla f(\mathbf{x}) = 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i \mathbf{x} + 2\mu \mathrm{Q}^H \mathrm{Q}\mathbf{x} - 2\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathbf{m}_i. \tag{4.12}$$

Therefore, the solution of regularized model can be obtained by substituting Eq. 4.12 as the gradient value of $f(\mathbf{x})$ in previously derived FISTA update equations.

4.1.2. Preconditioned FISTA

The main objective of the project is to implement a preconditioner within FISTA framework such that it reduces the reconstruction time of MRI. So, to include the preconditioner with FISTA, the first step of the algorithm given by Eq. 4.7 is reformed as

$$\mathbf{x}_k = \mathscr{T}_\alpha \left(\mathbf{x}_{k-1} - \frac{1}{L} M(A\mathbf{x} - \mathbf{b}) \right) \tag{4.13}$$

where, M is any suitable preconditioner. Here, the preconditioner is included in the gradient step of FISTA as this particular step is the bottleneck of the problem.

4.2. Structure of the System Matrix A

As stated in chapter 3, section 3.2 that the preconditioner must approximate the inverse of the system matrix, A, in some sense, thus, to find an appropriate preconditioner it is important to learn about the structure and properties possessed by the system matrix. System matrix, A, for the PI-CS model (Eq. 4.1) is described as

$$A = \sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i \tag{4.14}$$

where, R is a real matrix. Since R is real, $\mathrm{R}^H\mathrm{R}$ can be replaced by R alone. Analysing the Eq. 4.14, it can be inferred that $\mathrm{F}^H\mathrm{RF}$ is a BCCB matrix. This can be deduced from the diagonalisation property mentioned in section 3.2.2 that if C is the given BCCB matrix then it can be diagonalised by two-dimensional fourier transform as

$$D_1 = \mathrm{F}C\mathrm{F}^H \quad \text{or} \quad D_2 = \mathrm{F}^H C\mathrm{F},$$

conversely, if D_1 or D_2 is the given diagonal matrix then multiplication of two-dimensional Fourier transform matrices with diagonal matrix results in the BCCB matrix, C, shown below.

$$C = \mathrm{F}^H D_1 \mathrm{F} \quad \text{or} \quad C = \mathrm{F} D_2 \mathrm{F}^H$$

This is because the two-dimensional Fourier matrix, F, is a unitary matrix i.e. $\mathrm{FF}^H = I$. The BCCB structure of the term $\mathrm{F}^H\mathrm{RF}$ is depicted in Fig. 4.1. Here, an example of a relatively small dimension image (32×32) is considered for which the matrix $\mathrm{F}^H\mathrm{RF}$ has the dimensions 1024×1024. The BCCB structure of $\mathrm{F}^H\mathrm{RF}$ is lost when the matrices S_i and S_i^H are included.

It is observed that when $r = 1$ i.e. the k-space is fully sampled, A does not possess any structure except that its main diagonal entries are same. This can be visualized in the Fig. 4.2a. The fact that A has same main diagonal entries can be reasoned to the normalization of coil sensitivity maps i.e. the diagonal coil sensitivity matrix of each coil, S_i, is normalized by the root sum of squares of all the coil sensitivity map such that

$$\hat{\mathrm{S}}_i = \left(\sum_{k=1}^{N_c} \mathrm{S}_k^H \mathrm{S}_k \right)^{-\frac{1}{2}} \mathrm{S}_i. \tag{4.15}$$

when $r > 1$, A possesses a BCCB-like structure as shown in Fig.s 4.2b, 4.2c and 4.2d. The prominence of this faded-BCCB structure increases with increase in r which can be seen in Fig. 4.2, where structure of A for $r = 8$, Fig. 4.2d, has more prominent BCCB-like structure than A with $r = 4$, Fig. 4.2c, which in turn has

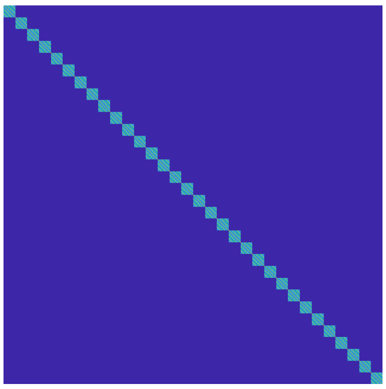


Figure 4.1: BCCB structure of matrix $\mathrm{F}^H\mathrm{RF}$.

more prominent structure than A with $r = 2$, Fig. 4.2b. Therefore, lesser the measured k-space lines, more prominent is the BCCB-like structure. In practice, A is not exactly a BCCB matrix because its main diagonal blocks are different whereas in case of a BCCB matrix the diagonal blocks are same.

Another interesting property of A is that it is a hermitian matrix. This is apparent from Eq. 4.14 because A^H results in the following

$$A^H = \left(\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i\right)^H = \sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i = A.$$

Thus, A is a hermitian matrix with the same diagonal elements. However, for the regularized problem formulation A is described as

$$A_{\text{reg}} = \sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i + \mu \mathrm{Q}^H \mathrm{Q} \tag{4.16}$$

for which the main diagonal entries are different because of addition of the $\mathrm{Q}^H\mathrm{Q}$ term which is a diagonal matrix containing the information about the background. Yet, the regularized system matrix is a hermitian matrix and possess the same BCCB-like structure for $r > 1$.

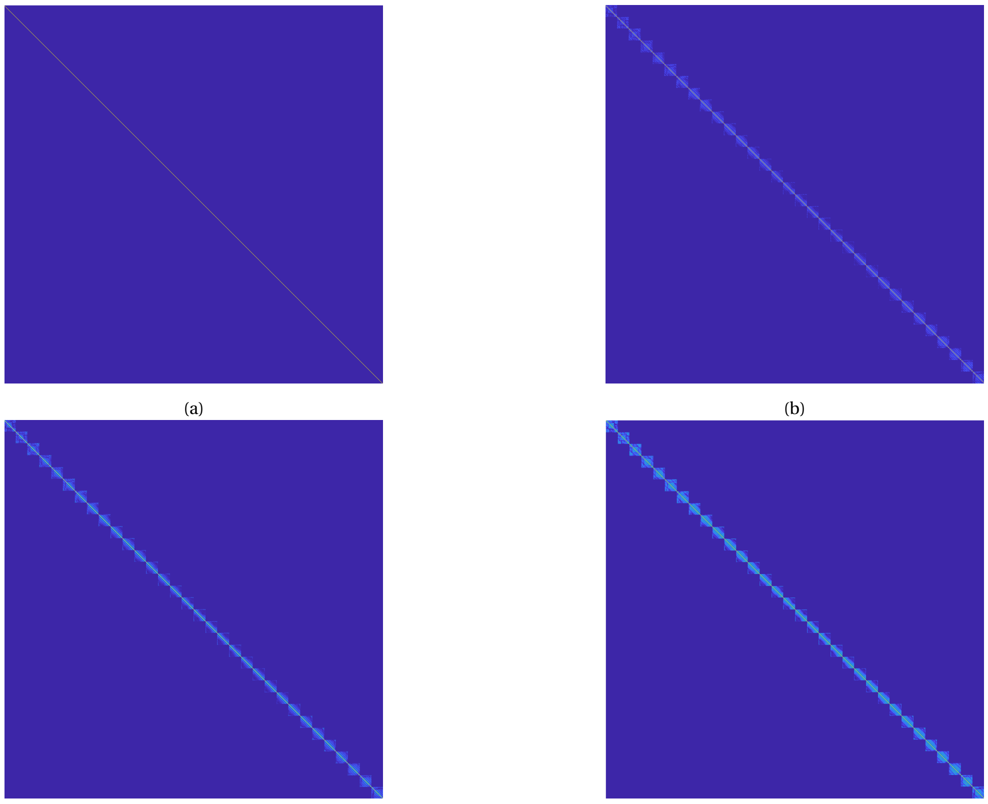
4.3. Implementation of Existing Preconditioners

To begin with the search for appropriate preconditioners, the study implemented two existing preconditioners namely Jacobi and circulant preconditioner explained in section 3.2.1 and 3.2.2 respectively. The implementation of the same for the current model is discussed in this section.

4.3.1. Jacobi Preconditioner

Jacobi's preconditioner requires to find the diagonal elements of A. Considering the unregularized PI-CS model for which A is defined as in Eq. 4.14, the diagonal can be found by exploiting the structure of A. It can be recalled that A comprises of the $\mathrm{F}^H\mathrm{RF}$ term which is a BCCB matrix. BCCB matrices consist of circulant blocks where the blocks are arranged in a circular manner. For instance, if C is a BCCB matrix of dimension $N \times N$, it has the following structure

$$C = \begin{bmatrix} C_0 & C_{N-1} & \cdots & C_2 & C_1 \\ C_1 & C_0 & C_{N-1} & \cdots & C_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ C_{N-2} & \ddots & C_1 & C_0 & C_{N-1} \\ C_{N-1} & C_{N-2} & \cdots & C_1 & C_0 \end{bmatrix} \tag{4.17}$$



(a) (b) (c) (d)

Figure 4.2: Structure of the system matrix A for brain with an image dimension 32×32. (a) Structure of A with $r = 1$ depicting same main diagonal entries. (b) Structure of A with $r = 2$. (c) Structure of A with $r = 4$. (d) Structure of A with $r = 8$.

where, C_j for $j = 1, \dots, N-1$ are circulant matrices with dimension $n \times n$ that have the following structure

$$C_j = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-2} & \ddots & c_1 & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix} \tag{4.18}$$

where, c_k for $k = 1, \dots, n-1$ are elements of the circulant matrix C_j. The circulant matrices, C_j, are special type of toeplitz matrices where not only the diagonal entries are same but also each row is shifted one element to right from the previous row. Due to the presence of these circulant blocks, BCCB matrix C has same main diagonal entries. This means $\mathrm{F}^H\mathrm{RF}$ has the same main diagonal entries. To obtain A, this term is multiplied by diagonal matrices S_i and S_i^H which destroys the property of same diagonal elements. However, due to the normalization of coil sensitivity maps A retains the property of same diagonal elements and its diagonal elements are in fact the diagonal elements of $\mathrm{F}^H\mathrm{RF}$. Proof of the same follows.

Let the BCCB matrix $\mathrm{F}^H\mathrm{RF}$ be denoted by C, then A can be written as

$$A = \sum_{i=1}^{N_c} \hat{\mathrm{S}}_i^H \underbrace{\mathrm{F}^H\mathrm{RF}}_{C} \hat{\mathrm{S}}_i$$

The diagonal entries of A arranged in column vector $\mathbf{a}$ can be obtained as

$$\mathbf{a} = \sum_{i=1}^{N_c} \hat{\mathbf{s}}_i^H \circ \mathbf{c} \circ \hat{\mathbf{s}}_i{}^T \tag{4.19}$$

where, $\hat{\mathbf{s}}_i^H$ is the column vector containing the diagonal entries of $\hat{\mathrm{S}}_i^H$ and $\mathbf{c}$ is the column vector containing the diagonal entries of C. Here, since, $\mathbf{c}$ has same entries because of BCCB structure, $\mathbf{c}$ can be replaced by its first element c_0. Therefore, Eq. 4.19 can be rewritten as

$$\mathbf{a} = \sum_{i=1}^{N_c} \hat{\mathbf{s}}_i^H \circ c_0 \circ \hat{\mathbf{s}}_i^T.$$

The constant term can be taken out of the summation

$$\mathbf{a} = c_0 \sum_{i=1}^{N_c} \hat{\mathbf{s}}_i^H \circ \hat{\mathbf{s}}_i^T. \tag{4.20}$$

Normalization of coil sensitivity maps, Eq. 4.15, can be written in terms of vectors $\mathbf{s}_i^H$ and $\mathbf{s}_i^T$, containing the diagonal elements of S_i^H and S_i, respectively, as the following

$$\hat{\mathbf{s}}_i = \frac{\mathbf{s}_i}{\sqrt{\sum_{k=1}^{N_c} \mathbf{s}_k^H \circ \mathbf{s}_k^T}}. \tag{4.21}$$

Substituting Eq. 4.21 in 4.20 results in

$$\mathbf{a} = c_0 \sum_{i=1}^{N_c} \frac{\mathbf{s}_i^H}{\sqrt{\sum_{k=1}^{N_c} \mathbf{s}_k^H \circ \mathbf{s}_k^T}} \circ \frac{\mathbf{s}_i^T}{\sqrt{\sum_{k=1}^{N_c} \mathbf{s}_k^H \circ \mathbf{s}_k^T}}$$

$$\mathbf{a} = c_0 \frac{\sum_{i=1}^{N_c} \mathbf{s}_i^H \circ \mathbf{s}_i^T}{\sum_{k=1}^{N_c} \mathbf{s}_k^H \circ \mathbf{s}_k^T} \tag{4.22}$$

$$\mathbf{a} = c_0 \mathbf{1} \tag{4.23}$$

where, $\mathbf{1}$ is an all ones vector. Hence, indeed the diagonal elements of A are same as the diagonal elements of C.

Now, to find the diagonal elements of A, it is required to find the diagonal elements of $C = \mathrm{F}^H\mathrm{R}\mathrm{F}$ which can be easily found using the properties of BCCB matrices. Ones that are particularly interesting for this purpose is that BCCB matrices are completely described by their first row or first column and that they are diagonalised by Fourier basis. For the given BCCB matrix, C, whose first column is denoted by $\mathbf{c}_{.,1}$ the following property holds

$$\mathbf{r} = \mathrm{F}\mathbf{c}_{:1}$$

where, $\mathbf{r}$ denotes the vector containing diagonal entries of R. In the present scenario, $\mathbf{r}$ is already known, thus, $\mathbf{c}_{:1}$ can be found as

$$\mathbf{c}_{:1} = \mathrm{F}^H\mathbf{r} \tag{4.24}$$

It is important to note that if C is defined as $\mathrm{F}\mathrm{R}\mathrm{F}^H$ then the BCCB matrix is defined by its first row and in that case $\mathbf{r} = \mathrm{F}\mathbf{c}_{1:}$ where $\mathbf{c}_{1:}$ denotes the first row of C. Having found the $\mathbf{c}_{:1}$, its first element, $c_{11} = c_0$ is the diagonal entry of C. Therefore, the diagonal elements of C and A arranged in the vector form is given by the following equation

$$\mathbf{a} = c_{11}\mathbf{1}. \tag{4.25}$$

The diagonal entries of Jacobi preconditioner defined by Eq. 3.18 can then be found in the vector $\mathbf{p}$ as

$$\mathbf{p} = \mathbf{1} \oslash c_{11}\mathbf{1} \tag{4.26}$$

where, $\oslash$ denotes the symbol for element-wise division. Results for Jacobi preconditioned FISTA are depicted in section 7.2.1.

4.3.2. Circulant Preconditioner

Circulant preconditioners are a desirable choice because of its cheap construction and efficient implementation. These are particularly used when the system matrix has BCCB structure or can be approximated by a BCCB matrix. Since, the system matrix for the current model has a BCCB-like structure which motivates the choice of approximating A by a BCCB matrix. This idea was implemented in [6] for SENSE reconstruction with Cartesian trajectories in the split Bregman framework. To investigate the performance, this circulant preconditioner is implemented within FISTA framework. Choice of this particular circulant preconditioner is motivated by similar problem models. Its implementation is straightforward from the explanation in section 3.2.2.

5

Derivation of Preconditioner

This chapter dderives two new preconditioners for FISTA framework. It also discusses how the difficulties encountered during the design of the new preconditioners were overcome.

5.1. Polynomial Preconditioner

To further explore the field of preconditioner especially in the context of the FISTA framework, techniques that directly approximate the inverse of the system matrix, A, are considered. A classic example in the literature is the method of Frobenius norm minimization to approximate the inverse of sparse matrices [19]. The main aim of the method is to find a sparse matrix M that approximates the inverse of the given matrix A. This is formulated by the following minimization problem

$$\min_{M} \|I - MA\|_F^2 \tag{5.1}$$

where, $\|.\|_F$ denotes the Frobenius norm of the argument matrix. Here, M is the left preconditioner for matrix A. It is also possible to find the right preconditioner by minimizing the objective function, $\|I - AM\|_F^2 = \|I - M^H A^H\|_F^2$, which is equivalent to finding a left preconditioner for A^H. Therefore, if the system matrix is hermitian, the left and right preconditioner will be the same. Solving the least squares problem in Eq. 5.1 requires gradient type iterative methods. Here, the global minimal residual descent algorithm is considered to solve the minimization problem 5.1. Since, the method aims to find a sparse matrix, matrix elements below a specified tolerance are replaced with zero. This is known as the numerical dropping step where the elements with very small magnitudes are replaced with zero to obtain a sparse solution. The algorithm is demonstrated in Algo. 4.

Algorithm 4: Global Minimal Residual Descent Algorithm

```
1 initialize M, tol
2 while stopping criteria > tolerance do
3     $C := MA$;
4     $G := I - C$;
5     $\alpha = \mathrm{tr}(G^H AG)/\|AG|_F^2$;
6     $M := M + \alpha G$;
7     $M(|M| < tol) = 0$;
8 end
```

To study if solving the minimization problem 5.1 using algorithm 4 produces the desired result, a simple example with smaller image dimensions, 64×70, was considered. The system matrix corresponding to this example was relatively small (4480×4480) and could be stored explicitly. Using algorithm 4 with normalised residual Frobenius norm error as the stopping criteria, defined as

$$e_k = \frac{\|I - M_k A\|_F}{\|I\|_F} \tag{5.2}$$

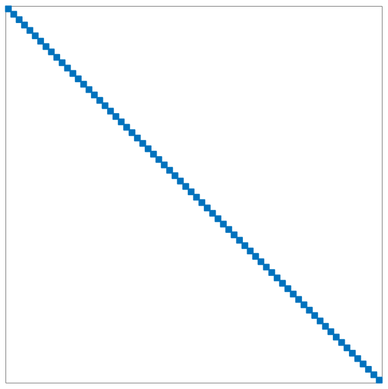


Figure 5.1: Sparsity pattern of preconditioner M for smaller dimensional problem.

where, e_k represents the error at k^{th} iteration, for the smaller dimensional problem results in M that was found in about 43 iterations with a residual error of $9.013e-02$. Using this M as the preconditioner in regularized FISTA framework reduced the condition number of the system matrix from 1.475e+03 to 2.80 which means M indeed provides a good approximation of the inverse of A. The obtained M is sparse with elements only on the diagonal blocks as represented in Fig. 5.1. The preconditioned FISTA for the smaller dimensional problem converges in significantly lesser number of iterations compared to FISTA alone. However, it comes with an expense of the overwhelming time to calculate the preconditioner, as performing matrix-matrix product in the algorithm 4 is extremely expensive computation wise. Further, to implement algorithm 4, A and M should be stored explicitly which for large problem size is not possible because of memory limitations. With lack of explicit system matrix and memory restrictions, it is impossible to find a preconditioner and even if one can obtain such a preconditioner, it is computationally inefficient. Apart from this, the system matrix is only available as a callback function that can be evaluated on a vector which makes the implementation of the algorithm 4 redundant.

Table 5.1: Condition numbers of various system matrices for problem size of 64×70 with undersampling factor $r = 2$

S.No.	**System Matrix**	**Condition Number**
1.	A	1.475e+03
2.	$M_{45}A$	17.93
3.	M_2A	464.68
4.	A_{reg}	217.477
5.	$M_{45}A_{reg}$	2.8
6.	M_2A_{reg}	63.38

Therefore, to implement this preconditioner for the current model it is important to address the drawbacks stated above. First, considering the time consumed by algorithm 4 to calculate the preconditioner, it is observed that with the specified stopping criteria (Eq. 5.2) and $tol = 10e-04$, M is obtained at 45^{th} iteration. To check how well it approximates the inverse of A, the condition number of $M_{45}A$ is computed which comes out to be 17.93. For reference, closer the condition number is to 1 better the problem is posed. Comparing the condition number before and after the preconditioning matrix M, it is observed that M_{45} is indeed a good approximation of inverse of A. However this comes at an expense of the construction time of M_{45}. One way to overcome this is by using the M at 2^{nd} iteration of algorithm 4 instead of 45^{th}. Referring to the table 5.1, it can be seen that condition number of system matrix A with M_2 preconditioner is far less than the condition number of A. Therefore, M_2 is a suitable candidate as its construction time is negligible compared to M_{45}.

Second, due to memory limitations and huge problem size it is only possible to evaluate A in an operator form on a vector. This can be addressed by expanding M_2 to write it in terms of the initial matrix M_0 and then implementing it as an operator on a vector. To write M_2 in the expanded form, M_k is explicitly derived for first two or three iterations ($k = 1,2$ and 3) of algorithm 4 given by the following:

- **Iteration 1**

$$\begin{aligned} C_1 &= M_0 A \\ G_1 &= I - C_1 \\ \alpha_1 &= \frac{\mathrm{tr}(G_1^H A G_1)}{\mathrm{tr}(G_1^H A^H A G_1)} \\ M_1 &= M_0 + \alpha_1 (I - M_0 A) \end{aligned} \tag{5.3}$$

- **Iteration 2**

$$\begin{aligned} C_2 &= M_1 A = [M_0 + \alpha_1 (I - M_0 A)] A \\ G_2 &= I - C_2 \\ \alpha_2 &= \frac{\mathrm{tr}(G_2^H A G_2)}{\mathrm{tr}(G_2^H A^H A G_2)} \\ M_2 &= M_0 + (I - M_0 A)[\alpha_1 + \alpha_2 - \alpha_1 \alpha_2 A] \end{aligned} \tag{5.4}$$

- **Iteration 3**

$$\begin{aligned} C_3 &= M_2 A = \{M_0 + (I - M_0 A)[\alpha_1 I + \alpha_2 I - \alpha_1 \alpha_2 A]\} A \\ G_3 &= I - C_3 \\ \alpha_3 &= \frac{\mathrm{tr}(G_3^H A G_3)}{\mathrm{tr}(G_3^H A^H A G_3)} \\ M_3 &= M_0 + (I - M_0 A)[(\alpha_1 + \alpha_2 + \alpha_3) I - (\alpha_1 \alpha_2 + \alpha_1 \alpha_3 + \alpha_2 \alpha_3) A + \alpha_1 \alpha_2 \alpha_3 A^2] \end{aligned} \tag{5.5}$$

If the initial matrix M_0 is set as an all zero matrix, O, then the preconditioners at three iterations can be written as

$$M_1 = \alpha_1 I \tag{5.6a}$$

$$M_2 = (\alpha_1 + \alpha_2) I - \alpha_1 \alpha_2 A \tag{5.6b}$$

$$M_3 = (\alpha_1 + \alpha_2 + \alpha_3) I - (\alpha_1 \alpha_2 + \alpha_1 \alpha_3 + \alpha_2 \alpha_3) A + \alpha_1 \alpha_2 \alpha_3 A^2 \tag{5.6c}$$

Since all these preconditioners are polynomials in A, they can be implemented in an operator form. Thus, if the values of α_1, α_2 and α_3 can be estimated then M_2 and M_3 can be considered as potential preconditioner candidates for FISTA framework.

Estimating α_1, α_2 and α_3 is not straightforward because it requires the diagonals of the following matrix-matrix products: (i)$G^H AG$ and (ii) $G^H A^H AG$, which vary for each alpha. Further, because of the huge problem size, these terms are not explicitly available. So, in order to estimate these coefficients, additional assumptions are made based on the structure of system matrix. In section 4.2, it was observed that A has same main diagonal entries and that A is a hermitian matrix i.e. $A^H = A$. From the property of hermitian matrices, M should also be a hermitian matrix as it approximates the inverse of A. This further leads to C, G, $G^H AG$ and $G^H A^H AG$ being hermitian matrices as well. Now, assuming that the terms $G^H AG$ and $G^H A^H AG$ are hermitian matrices with same diagonal entries, then an approximation of the coefficient values can be made by calculating the first diagonal entry of these terms and multiplying it by the size of the matrix, represented by the following equation

$$\alpha_i \approx \frac{N \times [\text{First diagonal entry of } G_i^H A G_i]}{N \times [\text{First diagonal entry of } G_i^H A^H A G_i]} = \frac{\text{First diagonal entry of } G_i^H A G_i}{\text{First diagonal entry of } G_i^H A^H A G_i} \tag{5.7}$$

where, i represents the coefficient number corresponding to the iteration number at which it was introduced, given by Eq.s 5.3, 5.4 and 5.5. The diagonal entries for each iteration can be found as follows.

Let $\mathbf{m}_{:1}^0$, $\mathbf{c}_{:1}^1$ and $\mathbf{g}_{:1}^1$ represents the first column of matrix M^0, C^1 and G^1 respectively. Initialising M^0 as an all zero matrix, O, means that

$$\mathbf{m}_{:1}^0 = \mathbf{0}$$

where, $\mathbf{0}$ is an all zero column vector. Then, from Eq. 5.3, it follows that

$$\mathbf{c}_{:1}^1 = A\mathbf{m}_{:1}^0 \tag{5.8}$$

and

$$\mathbf{g}_{:1}^1 = \mathbf{e}_1 - \mathbf{c}_{:1}^1 \tag{5.9}$$

where, $\mathbf{e}_1$ is the first standard basis column vector. Having calculated the first columns of C and G, α_1 is given as

$$\alpha_1 = \frac{(\mathbf{g}^1_{:1})^H[A\mathbf{g}^1_{:1}]}{(\mathbf{g}^1_{:1})^H[A(A\mathbf{g}^1_{:1})]} \tag{5.10}$$

where, multiplications with A are performed in an operator form. Similarly, α_2 can be calculated which requires the computation of $\mathbf{m}^1_{:1}$ given as

$$\mathbf{m}^1_{:1} = \mathbf{m}^0_{:1} + \alpha_1(\mathbf{g}^1_{:1}). \tag{5.11}$$

These equations are calculated in an iterative manner to compute all the coefficients, represented by algorithm 5.

Algorithm 5: Coefficient Estimation

```
1 initialize coefficients = 3, $\mathbf{m}^0_{:1} = \mathbf{0}$
2 for $j$ = 1:coefficients do
3     $\mathbf{c}^j_{:1} := A\mathbf{m}^{j-1}_{:1}$;
4     $\mathbf{g}^j_{:1} := \mathbf{e}_1 - \mathbf{c}^j_{:1}$;
5     $\alpha_j := \frac{(\mathbf{g}^j_{:1})^H A\mathbf{g}^j_{:1}}{(\mathbf{g}^j_{:1})^H AA\mathbf{g}^j_{:1}}$;
6     $\mathbf{m}^j_{:1} := \mathbf{m}^{j-1}_{:1} + \alpha_j\mathbf{g}^j_{:1}$;
7 end
```

This approach estimate alphas based on the assumption that the diagonal elements of G^HAG and G^HA^HAG are same which is true in the present case because of normalised coil sensitivity maps. However, to build an extensive preconditioner more robust methods to estimate the values of alpha are required.

In search of better methods to optimize the coefficient values of the polynomial preconditioner, Chebyshev's method to estimate the coefficients of the polynomial was explored. Since, the preconditioners given in Eq. 5.6b and 5.6c are polynomials in A similar to Chebyshev's polynomial preconditioner [19], the coefficients, α_j, can be calculated in the same manner as coefficients of the Chebyshev's polynomial of first kind. The criteria to find the coefficients of the Chebyshev's polynomial preconditioner, $s(A)$, is to find a particular $s(A)$ that approximates the inverse of A by solving the minimization problem which aims to bring the spectrum of the preconditioned problem close to the spectrum of identity. This is formulated as finding a polynomial $s \in \mathbb{P}^k$, where $\mathbb{P}^k$ denotes the space of the polynomials with degree not exceeding k, that solves the following problem

$$\min_{s\in\mathbb{P}^k} \max_{\lambda\in[\alpha,\beta]} |1-\lambda s(\lambda)| \tag{5.12}$$

where $\max_{\lambda\in[\alpha,\beta]} |1-\lambda s(\lambda)|$ denotes the spectral radius of the preconditioned system, $As(A)$, and $[\alpha,\beta]$ denotes the interval containing the eigenvalues of A. Eq. 5.12 can also be written as

$$\min_{q\in\mathbb{P}^{k+1}} \max_{\lambda\in[\alpha,\beta]} |q(\lambda)| \tag{5.13}$$

Solving the problem 5.13 results in the following

$$q(t) = \frac{T_{k+1}\left(\frac{\beta+\alpha-2t}{\beta-\alpha}\right)}{T_{k+1}\left(\frac{\beta+\alpha}{\beta-\alpha}\right)}$$

where, T_{k+1} is a three term recursion given by the following equations

$$T_{k+1}(t) = 2tT_k(t) - T_{k-1}(t),$$

$$T_1(t) = t$$

and

$$T_0(t) = 1.$$

Polynomial $s(t)$ can then be found as the following

$$s(t) = \frac{1-q(t)}{t}.$$

To find the desired polynomial, it is important to know the interval in which the eigenvalues of the system matrix lies. Finding this interval is a challenging task especially because of the huge problem size and lack of an explicit system matrix. Assuming the interval to be $[0,1]$, the Chebyshev's polynomial for $k=1$ and $k=2$ is given by

$$p_1(t) = 8-8t \tag{5.14}$$

and

$$p_2(t) = 18-48t+32t^2 \tag{5.15}$$

respectively. Coefficients of the Chebyshev's polynomial did not provide good-estimates of the values of alphas as with these coefficient values, the algorithm starts to diverge. Hence, the idea to use Chebyshev's polynomial was dropped. Thus, the problem to find better methods to estimate the coefficient values remains an open problem which needs further investigation.

5.2. BLOCK DIAGONAL CIRCULANT PRECONDITIONER

Literature is the witness that circulant preconditioners are the favourable choice for large dimensional image reconstruction problems. This is because circulant matrices can be easily evaluated on vectors in an operator form because of their diagonalisation property. Here, a block diagonal circulant preconditioner is derived to approximate the inverse of A which is inspired by the work of Koolstra et al. [6]. Their work assumes A to be a BCCB matrix and its inverse is approximated using the diagonalisation property of BCCB matrices (described in detail in section 3.2.2). The resulting preconditioner from their approach has the structure as given by Eq. 4.18, indicating that the preconditioner has constant diagonal blocks. Using the small dimension problem (64×70), the condition number of preconditioned system matrix (with this BCCB preconditioner) comes out to be 916.82. Since the BCCB preconditioner reduces the condition number of the system matrix, it is a good inverse approximation of A.

Alternatively, it is rational to write A as a block diagonal matrix because this way each diagonal block is different unlike when assumed to be BCCB matrix. The block diagonal matrix will have the following structure.

$$A \approx \hat{A} = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_m \end{bmatrix} \tag{5.16}$$

Approximating each hermitian block by circulant block, $\hat{A}$ can be written as

$$\hat{A} = \begin{bmatrix} \mathrm{F}_n^H \underbrace{\mathrm{F}_n A_1 \mathrm{F}_n^H}_{K_1} \mathrm{F}_n & 0 & \cdots & 0 \\ 0 & \mathrm{F}_n^H \underbrace{\mathrm{F}_n A_2 \mathrm{F}_n^H}_{K_2} \mathrm{F}_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{F}_n^H \underbrace{\mathrm{F}_n A_m \mathrm{F}_n^H}_{K_m} \mathrm{F}_n \end{bmatrix} \tag{5.17}$$

where, F_n is n-point one-dimensional Fourier transform and K_j for $j \in \{1,\ldots,m\}$ are diagonal matrices. Since, in practice every block is hermitian than circulant so K_j's are not diagonal matrices. Yet, it is reasonable to use the diagonal of each K_j because each K_j gathers a large part of the important information of A_j on its diagonal. The idea behind this step is exactly same as that proposed by Koolstra et al. [6] that removing the off diagonal elements of K_j will better approximate the inverse of A_j than removing the off diagonal elements of A_j itself. Hence, the preconditioner, M, that approximates the inverse of $\hat{A}$ is given by

$$M = \begin{bmatrix} \underbrace{\mathrm{F}_n^H \operatorname{diag}\{\mathbf{k}_1\}^{-1}\mathrm{F}_n}_{M_1} & 0 & \cdots & 0 \\ 0 & \underbrace{\mathrm{F}_n^H \operatorname{diag}\{\mathbf{k}_2\}^{-1}\mathrm{F}_n}_{M_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \underbrace{\mathrm{F}_n^H \operatorname{diag}\{\mathbf{k}_m\}^{-1}\mathrm{F}_n}_{M_m} \end{bmatrix} \tag{5.18}$$

where, $\mathbf{k}_j$ represents the vector containing the diagonal of K_j and diag$\{\cdot\}$ places the elements of argument on the diagonal of a matrix. Each block, M_j, has the dimension $n \times n$ for $j \in \{1, \ldots, m\}$. The major difference between this preconditioner and the one suggested by Koolstra et al. is that here all the (main) diagonal blocks are different whereas in their preconditioner all the (main) diagonal blocks are same. The condition number of preconditioned system matrix (with block diagonal circulant preconditioner) for the small dimension problem 64×70 comes out to be 400.62 which means that this preconditioner better conditions the system matrix. Condition numbers for system matrix and preconditioned system matrix (both BCCB preconditioner and block diagonal circulant preconditioner) are tabulated in table 5.2.

Table 5.2: Condition numbers of system matrix and circulant preconditioned system matrices for problem size of 64×70 with undersampling factor $r = 2$

S.No.	**System Matrix**	**Condition Number**
1.	A	1.475e+03
2.	$M_{\text{BCCB}}A$	916.82
3.	$M_{\text{BDC}}A$	400.62

It is important to realize the preconditioner in an operator form for its efficient implementation. This can be achieved in the similar way as done by Koolstra et al. [6], except that here the preconditioner is implemented in a block wise manner. The undersampling diagonal matrix, R, and the i^{th} coil sensitivity matrix, S_i in block wise manner are represented as

$$\mathrm{R} = \begin{bmatrix} \mathrm{R}_1 & 0 & \cdots & 0 \\ 0 & \mathrm{R}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{R}_m \end{bmatrix} \tag{5.19}$$

and

$$S_i = \begin{bmatrix} S_{i,1} & 0 & \cdots & 0 \\ 0 & S_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_{i,m} \end{bmatrix} \tag{5.20}$$

respectively. Here, R_j and S_j for $j \in \{1, \ldots, m\}$ are diagonal matrices of dimension $n \times n$. The diagonal blocks A_j for $j \in \{1, \ldots, m\}$ of $\hat{A}$ (Eq. 5.16) are then given as the following

$$\hat{A} = \begin{bmatrix} \sum_{i=1}^{N_c} \mathrm{S}_{i,1}^H \mathrm{F}_n^H \mathrm{R}_1 \mathrm{F}_n \mathrm{S}_{i,1} & 0 & \cdots & 0 \\ 0 & \sum_{i=1}^{N_c} \mathrm{S}_{i,2}^H \mathrm{F}_n^H \mathrm{R}_2 \mathrm{F}_n \mathrm{S}_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{i=1}^{N_c} \mathrm{S}_{i,m}^H \mathrm{F}_n^H \mathrm{R}_m \mathrm{F}_n \mathrm{S}_{i,m} \end{bmatrix}. \tag{5.21}$$

Note that the R is undersampling matrix where the matrix undersamples the data only in one direction, thus all the blocks of R are equal i.e. $\mathrm{R}_1 = \mathrm{R}_2 = \cdots = \mathrm{R}_m$. Now, to compute matrix M, the diagonals of K_j for

$j \in \{1,\dots,m\}$ are required. Let K be the matrix with K_j as its diagonal blocks, then from Eq. 5.17 and 5.21, K_j's in elaborated form are represented by the following equation

$$K = \begin{bmatrix} \sum_{i=1}^{N_c} \mathrm{F}_n \mathrm{S}_{i,1}^H \mathrm{F}_n^H \mathrm{R}_1 \mathrm{F}_n \mathrm{S}_{i,1} \mathrm{F}_n^H & 0 & \cdots & 0 \\ 0 & \sum_{i=1}^{N_c} \mathrm{F}_n \mathrm{S}_{i,2}^H \mathrm{F}_n^H \mathrm{R}_1 \mathrm{F}_n \mathrm{S}_{i,2} \mathrm{F}_n^H & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{i=1}^{N_c} \mathrm{F}_n \mathrm{S}_{i,m}^H \mathrm{F}_n^H \mathrm{R}_1 \mathrm{F}_n \mathrm{S}_{i,m} \mathrm{F}_n^H \end{bmatrix}. \tag{5.22}$$

Therefore, the diagonal elements, $\mathbf{k}_{i,j}$ for a certain coil, i ($i \in \{1,\dots,N_c\}$), and j^{th} ($j \in \{1,\dots,m\}$) block of K, can be found on the diagonal of $K_{i,j} = \underbrace{\mathrm{F}_n \mathrm{S}_{i,j}^H \mathrm{F}_n^H}_{C_{i,j}^H} \mathrm{R}_1 \underbrace{\mathrm{F}_n \mathrm{S}_{i,j} \mathrm{F}_n^H}_{C_{i,j}}$. Here, $C_{i,j} = \mathrm{F}_n \mathrm{S}_{i,j} \mathrm{F}_n^H$ is a circulant matrix. So, the diagonal elements $\mathbf{k}_{i,j}$ of $K_{i,j}$ can be found as

$$\mathbf{k}_{i,j} = \sum_{q=1}^{n} \mathbf{e}_q \left(\mathbf{c}_{q;i,j}^H \mathrm{R}_1 \mathbf{c}_{q;i,j} \right),$$

where, $\mathbf{c}_{q;i,j}^H$ is the q^{th} row of matrix $C_{i,j}^H$ and $\mathbf{e}_q$ is the q^{th} standard basis vector. As R_1 is a diagonal matrix, its diagonal elements in a vector form can be represented by $\mathbf{r}_1$. Subsequently, $\mathbf{k}_{i,j}$ can also be represented as

$$\begin{aligned} \mathbf{k}_{i,j} &= \sum_{q=1}^{n} \mathbf{e}_q \left(\mathbf{c}_{q;i,j}^H \circ \mathbf{c}_{qi,j}^T \right) \mathbf{r}_1 \\ &= \begin{bmatrix} \mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T \\ \mathbf{c}_{2;i,j}^H \circ \mathbf{c}_{2;i,j}^T \\ \vdots \\ \mathbf{c}_{n;i,j}^H \circ \mathbf{c}_{n;i,j}^T \end{bmatrix} \mathbf{r}_1 \\ &= \left(\mathrm{C}_{i,j}^H \circ \mathrm{C}_{i,j}^T \right) \mathbf{r}_1, \end{aligned} \tag{5.23}$$

where $\circ$ denotes the hadamard product. The multiplication $\mathrm{C}_{i,j}^H \circ \mathrm{C}_{i,j}^T$ contains product of two circulant matrices which is again a circulant matrix. Since, circulant matrices possess special structure, their evaluation on a vector can be represented as the circular convolution of first row/column of the circulant matrix and the vector. Therefore, Eq. 5.23 can be written as

$$\mathbf{k}_{i,j} = (\mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T)^T * \mathbf{r}_1. \tag{5.24}$$

Using the circular convolution theorem on above equation gives us

$$\mathrm{F}_n \mathbf{k}_{i,j} = \mathrm{F}_n \left[\left(\mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T \right)^T * \mathbf{r}_1 \right] = \mathrm{F}_n \left[\left(\mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T \right)^T \right] \circ \mathrm{F}_n \mathbf{r}_1.$$

As a result, the diagonal elements $\mathbf{k}_{i,j}$ can be found as

$$\mathbf{k}_{i,j} = \mathrm{F}_n^H \left\{ \left[\mathrm{F}_n \left(\mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T \right)^T \right] \circ \mathrm{F}_n \mathbf{r}_1 \right\} \tag{5.25}$$

where, the first row $\mathbf{c}_{1;i,j}^H$ of matrix $C_{i,j}^H$ is given by $(\mathbf{c}_{1;i,j}^H)^T = \mathrm{F}_n^H (\mathbf{s}_{i,j}^H)$ with $\mathbf{s}_{i,j}^H$ being the column vector containing the diagonal elements of $\mathrm{S}_{i,j}$. For multiple coils, diagonal elements of j^{th} block is given by

$$\mathbf{k}_j = \mathrm{F}_n^H \left\{ \left[\mathrm{F}_n \sum_{i=1}^{N_c} \left(\mathbf{c}_{1;i,j}^H \circ \mathbf{c}_{1;i,j}^T \right)^T \right] \circ \mathrm{F}_n \mathbf{r}_1 \right\}. \tag{5.26}$$

Eq. 5.26 can be efficiently computed in an operator form using the FFT operation. Further, matrix M can be efficiently computed in an operator form using two additional FFT's. The results of this preconditioner with FISTA are presented in chapter 7.

6 Computational Implementation

In this chapter, the computational implementation of the derived preconditioners (discussed in the previous chapter) in MATLAB® (9.4, R2018a) is discussed.

6.1. Data

The dataset used in the study to test the algorithms is the measurement of T_2-weighted scan of the 3D volume of the brain having 28 slices with a 3T MR system having 13-channel head coil for reception of signals. Thickness of each slice is 4mm. The data is acquired using the turbo spin-echo (TSE) sequence with the scan factor = 15 i.e. 15 k-space lines are acquired at the same time. The voxel size of the data is 0.4 and the field of view (FOV) is 23×23 cm^2. The acquired data is arranged as $m \times n \times N_c \times slices$.

6.2. FISTA

From algorithm 3, implementation of FISTA is fairly straightforward with two additional functions. One that calculates the matrix-vector product, $A\mathbf{x}$, and second that performs the wavelet transform operation, W$\mathbf{v}$.

- Matrix-vector product ($A\mathbf{x}$): To compute A on a vector as an operator, i.e.

$$A\mathbf{x} = \sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R} \mathrm{F} \mathrm{S}_i \mathbf{x}$$

  where $\mathbf{x}$ is the unknown image whose elements are placed in a vector. The dataset of coil sensitivity maps and the undersampling mask is of the dimension $m \times n$ which as per the mathematical model is arranged at the diagonal of matrices S_i and R, respectively, with dimensions $mn \times mn$. For example, if the data has the dimensions 320×470 then the matrices S_i and R will have the size in order of $10^5 \times 10^5$. Due to the huge size of matrices, it is not possible to store them as diagonal matrices. Therefore, the diagonal elements of these matrices are stored in a vector form i.e. $\mathbf{s}_i$ and $\mathbf{r}$ contain the diagonal elements of matrices S_i and R respectively.

  The Fourier matrix F can be replaced by the in-built MATLAB® two-dimensional Fourier transformation function `Y = fft2(X)`. Here, the function accepts input as a matrix and returns the output also in a matrix. Therefore, when performing multiplication of F on a vector, it is reshaped into a matrix. In the implementation, the normalized `fft2` function is used which is shown in the following listing.

```
function Y = fft2n(X)
    Y = (1/sqrt(numel(X)))*fft2(X);
end
```

  Similarly, the inverse two-dimensional Fourier transformation function `ifft2` is also normalized, shown in the listing below.

```
function Y = ifft2n(X)
    Y = sqrt(numel(X))*ifft2(X);
end
```

The matrix-vector product $A\mathbf{x}$ is then computed in the form of a function given by the following listing.

```
function Ax = matrix_vector_product(x)
    for c = 1:Nc
        Sx = s(c).*x;
        FSx = fft2n(reshape(Sx,m,n));
        FSx = reshape(FSx,[],1);
        RFSx = r.*FSx;
        FRFSx = ifft2n(reshape(RFSx,m,n));
        FRFSx = reshape(FRFSx,[],1);
        SFRFSx(:,:,c) = conj(s).*FRFSx;
    end
A_x = sum(SFRFSx,3);
end
```

- Wavelet transformation function ($W\mathbf{v}$): To transform a given image, $\mathbf{v}$, into the wavelet domain, a publicly available wavelet operator class written by Michael Lustig [16] is used for the implementation. The class implements the Daubechies wavelet transform of an image in an operator form where `Y = WV` computes the wavelet transform of V and `V = W'Y` reconstructs it back. The operator accepts square matrix as input where the dimensions of the matrix must be in the power of two. So, to implement this wavelet operator, given image, $\mathbf{v}$, is reshaped into a matrix which is padded with last row and last column of it such that the dimension of the matrix is the next largest power of two. The function that pads the matrix is shown in the following listing.

```
function Y = pad_matrix(X)
dim = max(size(X));
    if rem(log2(dim),1) == 0
        dim_new = dim;
    else
        dim_new = pow2(ceil(log2(dim)));
    end
Y = zeros(dim_new);
Y(1:size(X,1),1:size(X,2)) = X;
Y(1:size(X,1),size(X,2)+1:end) = repmat(X(:,size(X,2)),1,size(new_X,2)-size(X,2));
Y(size(X,1)+1:end,1:end) = repmat(new_X(size(X,1),:),size(new_X,1)-size(X,1),1);
end
```

When the image is reconstructed back in the image domain, the extra padded numbers are truncated.

6.3. POLYNOMIAL PRECONDITIONED FISTA

Implementation of polynomial precondition with FISTA uses the same algorithm with two additional functions. First, that estimates the coefficients i.e. α_1, α_2 and α_3 and second that computes the preconditioner matrix-vector product.

- Estimation of coefficients: Alphas can be obtained by the direct implementation of the algorithm 5. Having calculated the values of alphas, the coefficients can be computed from the combination of alphas. The combination of alphas has a particular pattern that the first coefficient is just the sum of all the α_j, the second coefficient is the sum of products of combinations of 2 alphas out of total alphas and the third coefficient is the sum of product of combinations of 3 alphas out of total alphas. This is shown in the listing below.

```
vec = 1:no_of_alpha
for k = 1:no_of_alpha
    combination_alpha{k} = combnk(vec,k);
end
coefficient = [];
```

```
for i = 1:length(combination_alpha)
    temp = cell2mat(combination_alpha(i));
    for j = 1:size(combination_alpha{i},1)
        index = temp(j,:);
        prod=1;
            for iter = 1: size(index,2)
                prod = prod*alpha(index(iter));
            end
            coefficient(j) = prod;
    end
    prod_sum(i) = sum(coefficient);
    coefficient = [];
end
```

- Preconditioner matrix-vector product: Including the polynomial preconditioner (for instance, M_2 with FISTA results in the change of gradient of $f(x)$ to the following

$$\nabla f(\mathbf{x}) = (\beta_1 I - \beta_2 A)A\mathbf{x} - (\beta_1 I - \beta_2 A)\mathbf{b}. \tag{6.1}$$

  where, $\beta_1 = \alpha_1 + \alpha_2$ and $\beta_2 = \alpha_1\alpha_2$. It can be inferred that including the polynomial preconditioner results in few additional matrix-vector products of A on vectors and simple algebraic operations which has a straightforward implementation from Eq. 6.1.

6.4. BLOCK DIAGONAL CIRCULANT PRECONDITIONED FISTA

Implementation of block diagonal circulant preconditioned uses the imeplementation of Sec. 6.2 with two additional functions. One that constructs the diagonal elements of each block of the preconditioner and second that evaluates the preconditioner vector product.

- Construction of diagonal elements: The diagonal elements of each block are computed from the straightforward implementation of Eq. 5.26. The Fourier matrix F_n is replaced by the in-built MATLAB® one-dimensional Fourier transformation function `y = fft(x)`. Here, the function accepts input as a vector and returns the output also as a vector. In the implementation the normalised `fft` function is used which is shown in the following listing.

```
        function y = fftn(x)
        y = (1/sqrt(numel(x)))*fft(x);
        end
```

  Similarly, the inverse one-dimensional Fourier transformation function `ifft` is also normalized, shown in the listing below.

```
        function y = ifftn(x)
        y = sqrt(numel(x))*ifft(x);
        end
```

  The diagonal elements are then constructed by the following listing.

```
    function k = diagonal_elements
        for coil = 1:Nc
            S_coil = csm(:,:,coil);
            S_coil_vec = S_coil(:);
            for block = 0:M-1
                S_coil_block = conj(S_coil_vec(block*N + 1:block*N + N,1));
                Col_block(block*N + 1:block*N + N,1,1) = ifft(S_coil_block);
            end
        Col(:,coil) = Col_block;
        end
        r = double(random_mask(:));
        Col_squared = sum(abs(Col).^2,2);

        for block = 0:M-1
```

```
            r_block = r(1:N,1);
            Col_squared_block = Col_squared(block*N + 1:block*N + N,1);
            k(block*N + 1:block*N + N,1) = ...
                ifft((fft(r_block).*fft(Col_squared_block));
        end
    end
```

- Block diagonal circulant preconditioner vector product: This function evaluates the block diagonal circulant preconditioner on a vector. Implementation of this evaluation can traced to the matrix vector product of M in Eq. 5.18 on a vector. The following listing explains the computational implementation.

```
    function Mx = block_preconditioner_vector_product(x)
        for block = 0:M-1
            x_block = x(block*N + 1:block*N + N,1);
            Fv = ifft(x_block);
            kFv = (k(block*N + 1:block*N + N,1).^-1).*Fv;
            FkFv = fft(kFv);
            Mx(block*N + 1:block*N + N,1) = FkFv;
        end

    end
```

7 Results and Discussion

This chapter consists of the simulations and results. First, it presents the reconstruction results for the regularized model with FISTA. Second, it shows the reconstruction results of existing and derived preconditioners followed by their computational complexity.

The dataset utilized in this project has the dimensions $320 \times 470 \times 13$ where the first two dimensions represent the image size and the third dimension represents the number of receiver coil channels. The volume of the brain for this dataset is divided into 28 slices. All the results are presented with slice 9 of the given dataset. Both FISTA and Preconditioned FISTA are evaluated at two acceleration factors i.e. $r = 2$ and $r = 4$. The given data is fully sampled in k-space which is undersampled by multiplying the masks shown in Fig. 7.1. All the results are obtained using the regularised FISTA model i.e. Eq. 4.2.

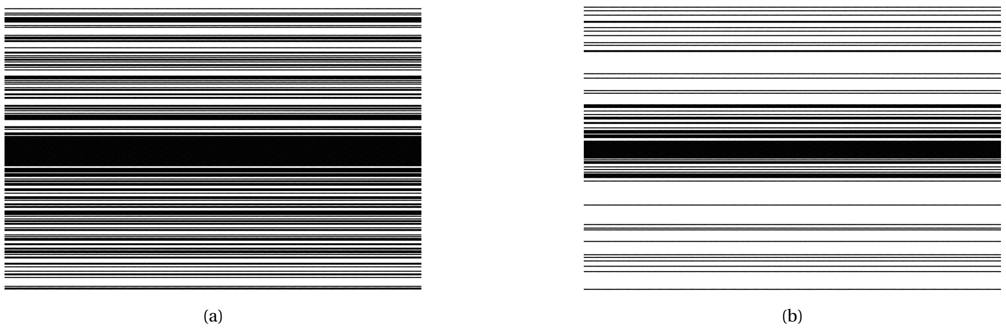


Figure 7.1: Undersampling masks for the given dataset. Black lines represent ones and white lines represent zeros. (a) Mask for $r = 2$. (b) Mask for $r = 4$.

7.1. Reconstruction using FISTA

The reconstructed results with FISTA for acceleration factor $r = 2$ and $r = 4$ are presented in Fig. 7.2. It represents the target image, the reconstructed image and the difference between the target image and reconstructed image. The difference is magnified with the factor of 3. These reconstructions are obtained by iterating FISTA for a fixed number of iterations.

In FISTA, the step size of the algorithm is controlled by the parameter 'L' which is the Lipschitz constant of $\nabla f(\mathbf{x})$ given by $2\sigma_{\max}(A)$. Here, $\sigma_{\max}(A)$ represents the largest singular value of A. The (nonzero) singular values of A are defined as the square roots of the nonzero eigenvalues of $A^H A$. Since, the system matrix is hermitian i.e. $A^H = A$, the singular and eigenvalues of A are closely related such that the nonnegative eigenvalue is also the singular value whereas the negative eigenvalue must reverse its sign to become the singular value. Here, we require only the largest singular value of A which by the above fact is equal to the largest eigenvalue of A. Hence, the Lipschitz constant L can be written as $2\lambda_{\max}(A)$.

The largest eigenvalue of the system matrix can not be computed in a straightforward manner because A is only available as a callback function that evaluates the matrix vector product. Therefore, an estimate of the largest eigenvalue of A is computed using the power iteration method. Since, the results are obtained using regularized FISTA, we find the largest eigenvalue of the regularised system A_{reg}. The estimate for the regularised system matrix is found to be $\lambda_{\text{max}}(A_{\text{reg}}) = 1.0996$. Accordingly, L is set to 2.

As the quality of image is significantly influenced by the regularization parameters, it is important to set them to provide the best image quality. Choice of ℓ_2 norm regularization parameter, μ, is based on the best visual representation of the image and the ℓ_1 norm regularization parameter, λ, is chosen such that the resulting image has both a reasonable signal to noise ratio (SNR) and perceived image quality. The plots representing the SNR of the reconstructed image with variation in λ is shown in Fig. 7.3. Hence, step size parameter and regularization parameters for these reconstructions are set as:

1. $r = 2 : L = 2$, $\lambda = 0.4$ and $\mu = 0.3$
2. $r = 4 : L = 2$, $\lambda = 0.4$ and $\mu = 0.3$

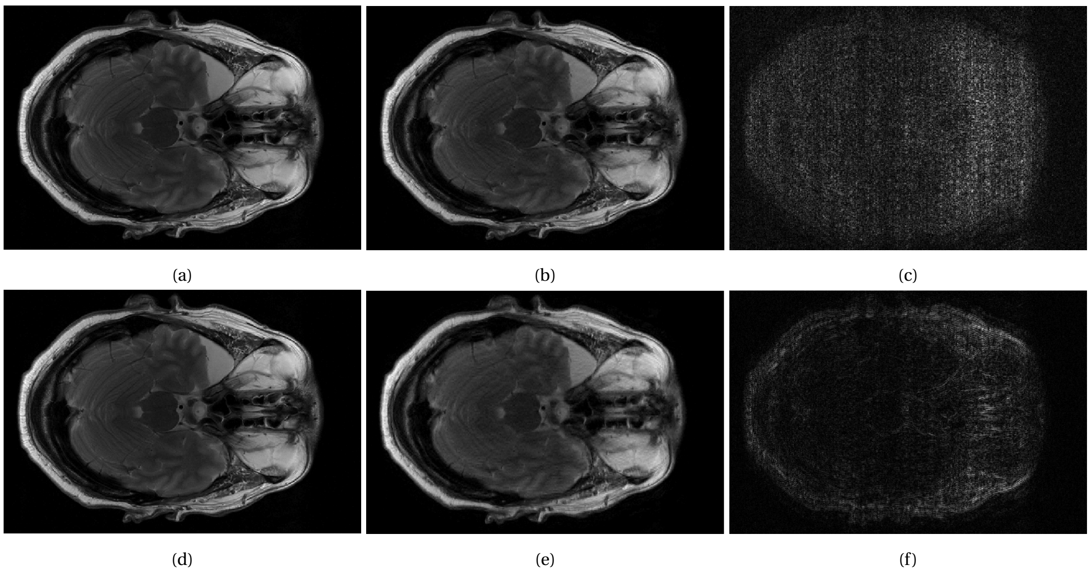


Figure 7.2: Reconstructed and difference image for given dataset for acceleration factor $r = 2$ and $r = 4$. (a) Target image. (b) Reconstructed image for $r = 2$. (c) Difference($\times 3$) image for $r = 2$. (d) Target image. (e) Reconstructed image for $r = 4$. (c) Difference($\times 3$) image for $r = 4$.

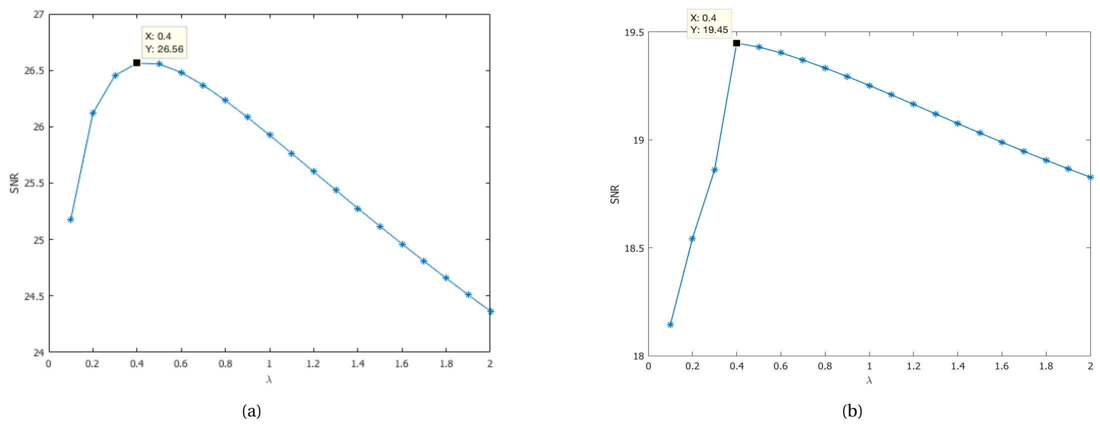


Figure 7.3: Analysis to find optimum λ. (a) λ for $r = 2$. (b) λ for $r = 4$.

7.2. Reconstruction using Preconditioned FISTA

This section presents the reconstruction results of preconditioned FISTA for Jacobi, circulant and derived preconditioners: polynomial and block diagonal circulant. The performance of these preconditioners is evaluated with two stopping criteria i.e. residual error and relative error. The stopping criteria are defined as the following.

1. **Residual Error**:
$$e_k = \frac{\|\mathbf{b} - A\mathbf{x}_k\|}{\|\mathbf{b}\|} \tag{7.1}$$
2. **Relative Error**:
$$e_k^{rel} = \frac{\|\mathbf{x} - \mathbf{x}_{\text{ref}}\|}{\|\mathbf{x}_{\text{ref}}\|} \tag{7.2}$$
where, $\mathbf{x}_{\text{ref}}$ is the reference image obtained from FISTA by keeping $r = 1$.

The tolerance for the specified error is set as

1. $r = 2$: (Residual error) $\epsilon = 8e^{-03}$ and (Relative error) $\epsilon_{\text{rel}} = 9e^{-02}$
2. $r = 4$: (Residual error) $\epsilon = 7e^{-03}$ and (Relative error) $\epsilon_{\text{rel}} = 13e^{-02}$

Preconditioning the system matrix A_{reg} improves its condition number, however, changes its eigenvalues. As the step size of FISTA depends on the maximum eigenvalue of the system matrix, the parameter 'L' for the preconditioned system matrices are set in accordance with their respective maximum eigenvalue.

7.2.1. Jacobi Preconditioner

The reconstruction image, difference image (magnified three times) and the error convergence plots for Jacobi preconditioned FISTA using the mentioned stopping criteria for $r = 2$ are shown in Fig. 7.4. The largest eigenvalue of the system matrix after including the preconditioner is $\lambda_{\text{max}} = 2.2845$, so, L is set to 4. Since L has an inverse relation with the step size, increasing the value of L reduces the step size of FISTA, thereby, making it slow. Thus, comparing the performance of the preconditioner with FISTA at same step size i.e. $L = 4$, it can be observed (Fig. 7.4e, 7.4f) Jacobi preconditioner does reduce the consumed iterations but it does not perform better than FISTA with $L = 2$. Under both stopping criteria, the number of iterations including and excluding the preconditioner are almost same when compared to FISTA with $L = 2$. At $r = 4$, the largest eigenvalue of the system matrix after including the preconditioner is $\lambda_{\text{max}} = 5.1965$, so L is set to 10. The results for $r = 4$ are presented in Fig. 7.5. The convergence for both the criteria at $r = 4$ follow the same trend as with $r = 2$.

Ideally, preconditioners with the diagonal structure are preferred because of their computational efficiency. However, in the current scenario, Jacobi preconditioner does not provide a good approximation of the inverse of A_{reg}. The reason can be owed to the constant diagonal elements of A (derived in 4.3.1) which are the result of coil sensitivity normalisation. Implying that the Jacobi preconditioner scales the system matrix with a constant value. Scaling the matrix provides a poor estimation for the system matrix inverse. Had the diagonal elements of A been different, Jacobi preconditioner would have provided a better approximation. Additionally, including the preconditioner increases the largest eigenvalue of the system matrix which makes the algorithm slower. Hence, it is meaningless to use this preconditioner for the current model.

7.2.2. Circulant Preconditioner

Reconstruction results using circulant preconditioner (derived in the work of [6]) with FISTA at $r = 2$ are depicted in Fig. 7.6. The largest eigenvalue of the system matrix after including circulant preconditioner is $\lambda_{\text{max}} = 10.3049$ and thus $L = 20$. Setting $L = 20$ implies that the circulant preconditioned FISTA utilizes a significantly small step size relative to FISTA, thereby making FISTA considerably slow. This can also be visualized by comparing the convergence plots of circulant preconditioned FISTA and FISTA at $L = 2$, shown in Fig. 7.6e and 7.6f. If the error convergence of circulant preconditioned FISTA is compared to FISTA with $L = 20$, then the reduced number of iterations is notable. In this case the former lower bounds the error convergence of latter.

For $r = 4$, reconstructed image, difference image and error convergence plots are presented in Fig. 7.7. Here, the largest eigenvalue of the system matrix after including circulant preconditioner is $\lambda_{\text{max}} = 45.2091$ and thus $L = 90$, implying that FISTA takes even small steps than at $r = 2$. Thus, making FISTA even slower which can be confirmed by the increase in consumed number of iterations where FISTA at $L = 2$ takes about 10 iterations, FISTA at $L = 20$ takes approximately 30 iterations and FISTA at $L = 90$ takes significantly more than 40 iterations. Even then, the error convergence trend at both the acceleration factors is similar to a great extent.

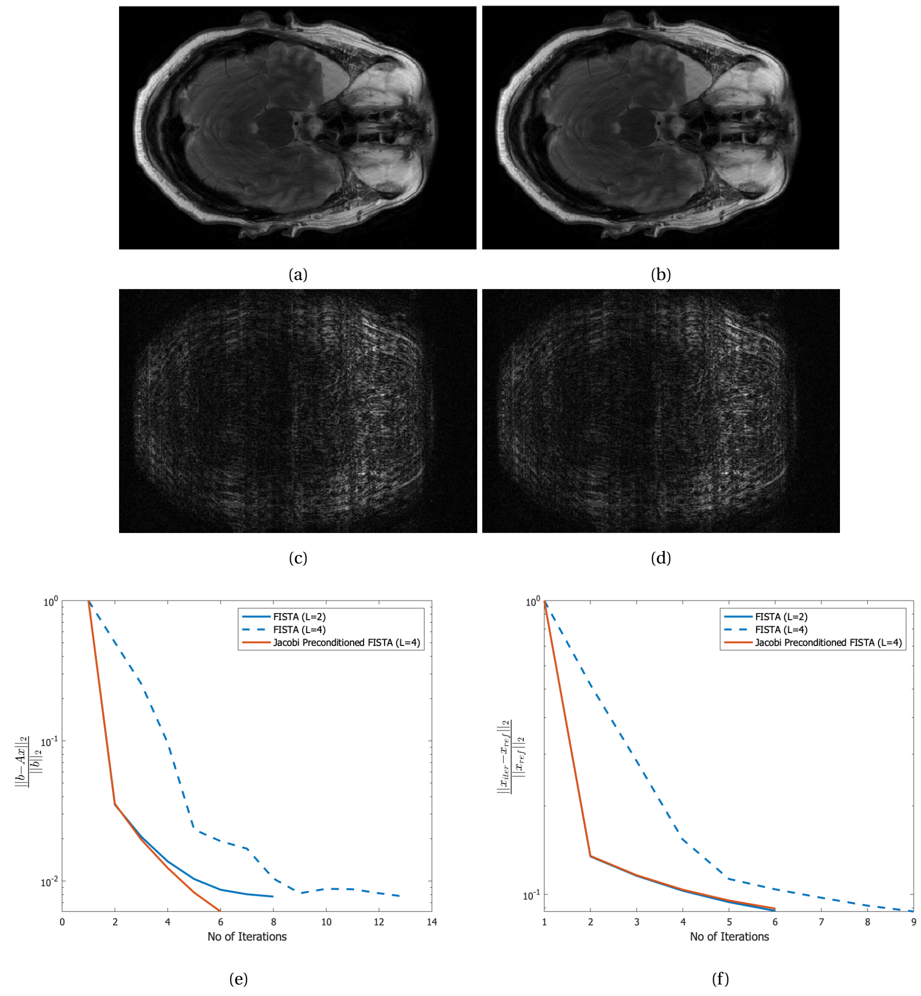


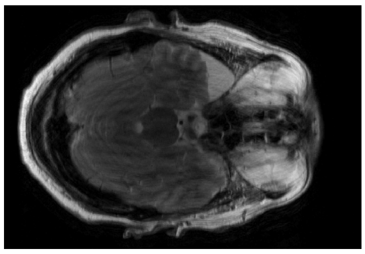
Figure 7.4: Reconstructed image, difference image and convergence plots for Jacobi preconditioned FISTA with $r = 2$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

A major drawback here is that the circulant preconditioner does not accurately approximate the inverse of the current system matrix A_{reg} as it does not take into account the diagonal matrix term $\mathrm{Q}^H\mathrm{Q}$. Even if it does, it is poor to approximate the inverse of the diagonal matrix $\mathrm{Q}^H\mathrm{Q}$ by a BCCB matrix. Undoubtedly, this preconditioner is computationally efficient, yet it is not a good choice for the current framework.

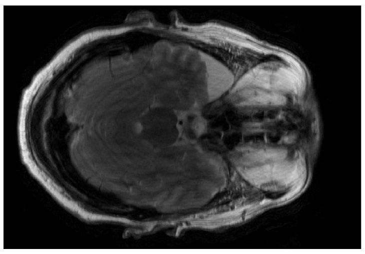
7.2.3. Polynomial Preconditioner

Section 5.1 derives two polynomial preconditioners i.e. M_2 and M_3. M_3 is a quadratic polynomial in A. Evaluation of M_3 on a vector will at least require six (Fast Fourier Transform) FFTs and additional linear operations. Alternatively, evaluation of M_3 on a vector can be interpreted as three operations of A on a vector, implying that the per iteration cost of M_3 preconditioned FISTA increases approximately by a factor of 3. This can be confirmed by comparing the number of flops consumed by FISTA and M_3 preconditioned FISTA which are computed in section 7.3. Therefore, this preconditioner is extremely expensive and practically infeasible for huge dimension problems.

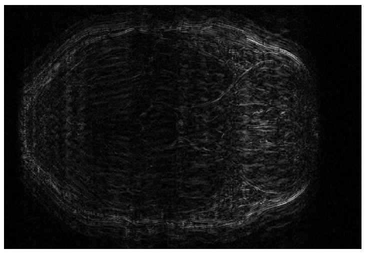
Reconstruction results with M_2 polynomial preconditioned FISTA at $r = 2$ are represented in Fig. 7.8. The largest eigenvalue of the system matrix after including the preconditioner is $\lambda_{\max} = 1.0001$, accordingly L is set to 2. An important point to note is that this preconditioner utilizes the same step size as FISTA. It can be observed from the error convergence plots (Fig. 7.8e, 7.8f) that M_2 preconditioned FISTA lower bounds the convergence of FISTA and the former takes approximately half the number of iterations to converge to the specified tolerance than the latter under both stopping criteria.



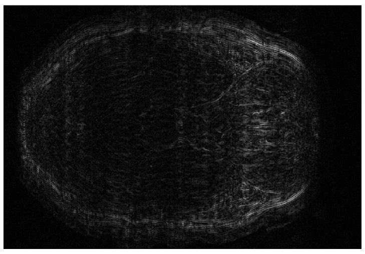
(a)



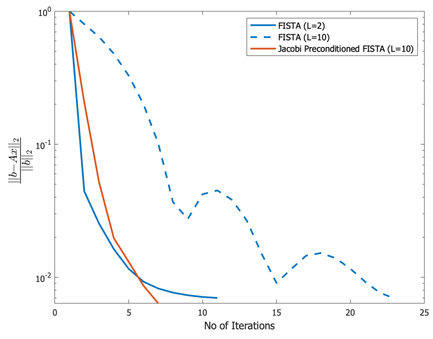
(b)



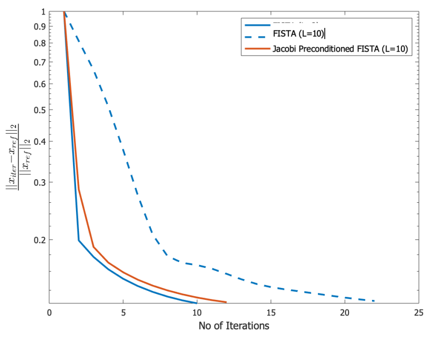
(c)



(d)



(e)



(f)

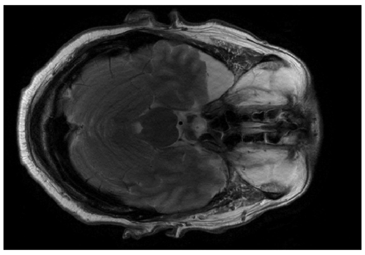
Figure 7.5: Reconstructed image, difference image and convergence plots for Jacobi preconditioned FISTA with $r = 4$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

Reconstructed image, difference image (magnified three times) and error convergence plots at $r = 4$ are depicted in Fig. 7.9. The largest eigenvalue of the preconditioned system matrix at $r = 4$ remains the same as at $r = 2$ and so L is set to 2. Even with increased acceleration factor, the preconditioned FISTA takes considerably fewer iterations than FISTA. Thus, M_2 preconditioned FISTA can be considered as a potential preconditioner for this framework.

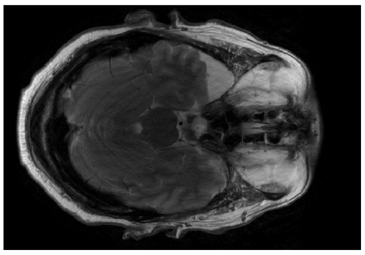
One of the limitations of M_2 preconditioner is that it only approximates the inverse of A whereas the actual system matrix is A_{reg}. To better approximate the inverse, the diagonal matrix term $\mathrm{Q}^H\mathrm{Q}$ should be taken into account. Due to time constraints, a detailed analysis of including the ℓ_2 norm term in the preconditioner was not completely investigated.

7.2.4. Block Diagonal Circulant Preconditioner

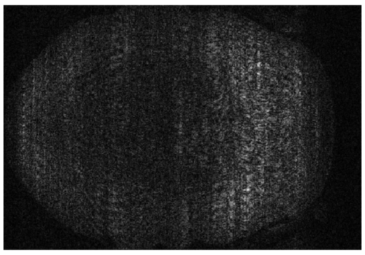
Fig. 7.10 represents the reconstruction image, difference image (magnified three times) and error convergence plots with block diagonal circulant preconditioned FISTA at $r = 2$. The largest eigenvalue of the preconditioned system matrix is $\lambda_{\max} = 14.3049$. Therefore, L is set to 28. Taking into account the inverse relationship of L and step size, setting $L = 28$ results in step size in the order of $4e^{-02}$. Similar to circulant preconditioner, this preconditioner also slowers the performance of FISTA by making its step size significantly low. This can be observed by comparing FISTA $L = 2$ with block diagonal circulant preconditioned FISTA in Fig. 7.10e and 7.10f. However, if the convergence of preconditioned FISTA is compared to FISTA $L = 28$ then indeed preconditioned FISTA performs better. Results for this preconditioner at $r = 4$ are presented in Fig. 7.11. At $r = 4$ the largest eigenvalues of the preconditioned FISTA is $\lambda_{\max} = 90.0561$, subsequently, $L = 180$.



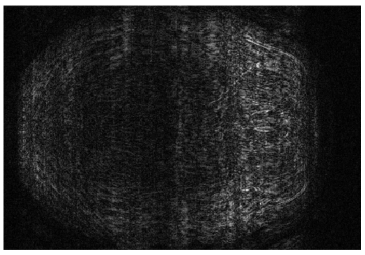
(a)



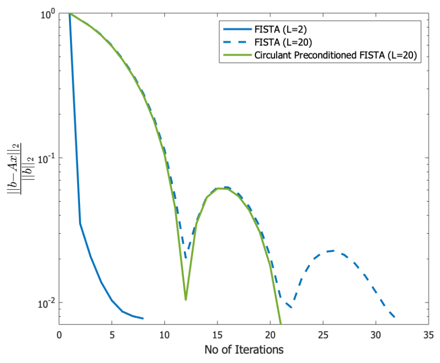
(b)



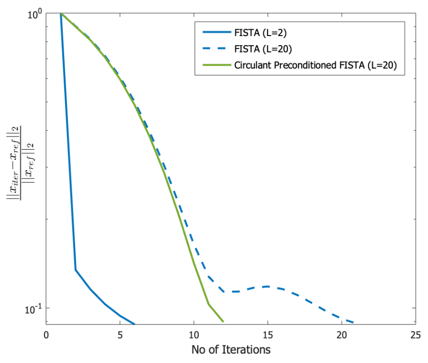
(c)



(d)



(e)



(f)

Figure 7.6: Reconstructed image, difference image and convergence plots for circulant preconditioned FISTA with $r = 2$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

The convergence behaviour of the preconditioner is almost the same at $r = 4$ as at $r = 2$.

The main drawback of the block diagonal circulant preconditioner is that it does not take into account the diagonal matrix term $Q^H Q$. Better inverse approximations can be achieved by incorporating the diagonal $Q^H Q$ matrix. Therefore, approximating the inverse of ℓ_2 norm regularized system matrix remains an open problem that needs further investigation.

In practice, preconditioners possessing special structures, for instance, circulant, BCCB, etc. are preferred because of their efficient implementations. However, for the current framework including efficient preconditioners comes with an expense of larger maximum eigenvalue. If the step size was independent of the largest eigenvalue of the system matrix, both circulant and block diagonal circulant preconditioner would be the obvious choice.

7.3. Computation Complexity of Derived Preconditioners

The section compares the derived preconditioners in the context of their computational complexity. It is measured by the number of Floating Point Operations (FLOPS) required by each preconditioner for its construction and evaluation on a vector. Tab. 7.1 lists the required flops for the derived preconditioners.

Table 7.1 reveals that construction of M_3 takes 1.5 times the number of flops used in the construction of M_2 and almost 9 times the number of flops required by evaluation of A on a vector. Moreover, its evaluation on a vector is approximately 3 times the flops required by A to be evaluated on a vector. This means that M_3 is not only expensive to construct but it is also expensive to be evaluated on a vector. Using such a preconditioner

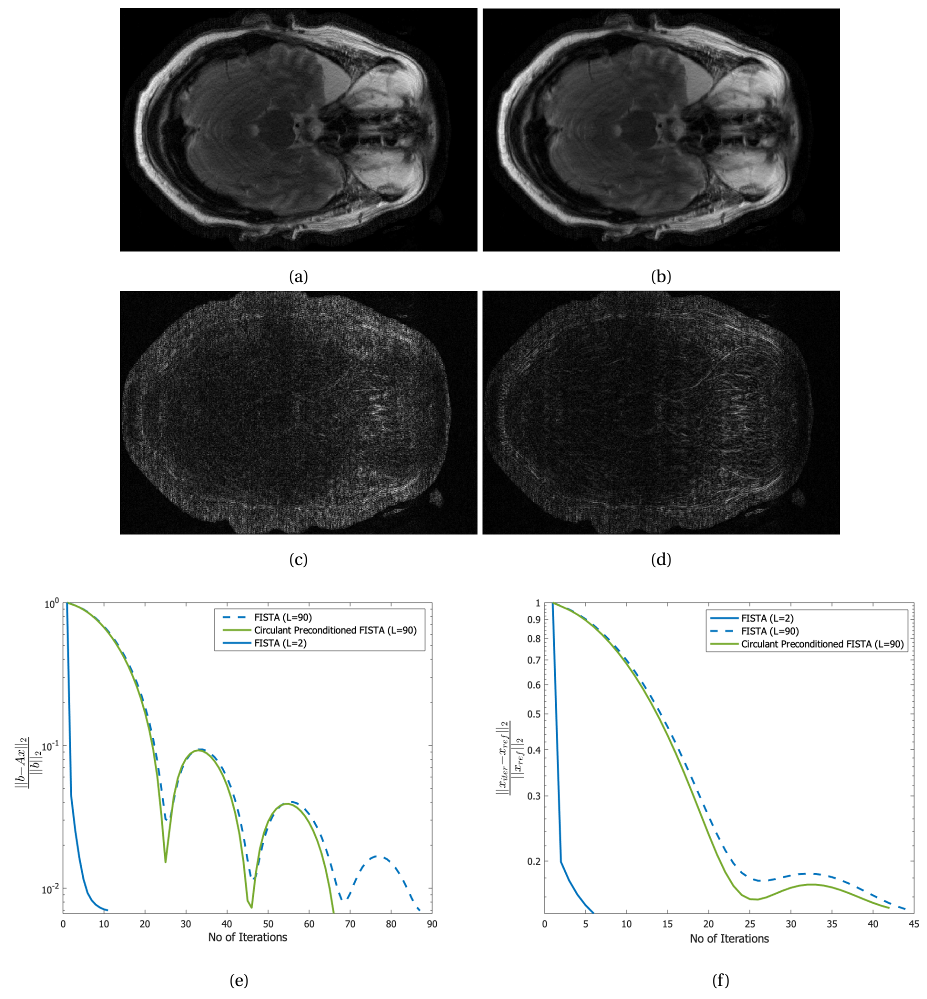
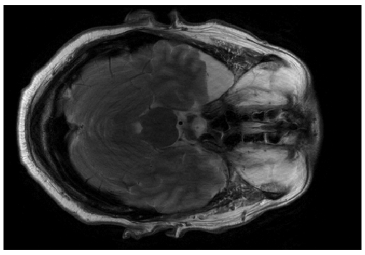

(a)
(b)
(c)
(d)
FISTA (L=90)
Circulant Preconditioned FISTA (L=90)
FISTA (L=2)
$\frac{||b-Ax||_2}{||b||_2}$
No of Iterations
$\frac{||x_{iter}-x_{ref}||_2}{||x_{ref}||_2}$
(e)
(f)

Figure 7.7: Reconstructed image, difference image and convergence plots for circulant preconditioned FISTA with $r = 4$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

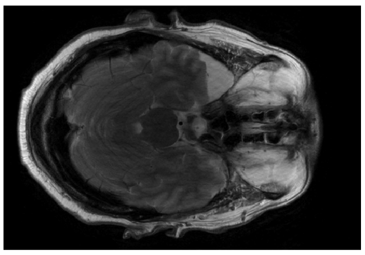
is trivial because it may happen that the amount of time reduced using this preconditioner will be equivalent to the time taken by the preconditioner's construction and evaluation, leaving us with zero gain. Therefore, it is impractical to use such an extremely expensive preconditioner.

Analysing the number of flops consumed by M_2 preconditioner, it can be noted that construction of M_2 takes 6 times the number of flops required by evaluation of A on vector indicating that construction of M_2 is not cheap either. Additionally, it can be interpreted that the required flops are directly proportional to the number of coils indicating that more the number of coils, more expensive will be the construction. Analysing the evaluation of M_2 preconditioner on a vector, it can be inferred that it costs (almost) the same number of flops as consumed by evaluation of A on a vector. In such a case if the overall time gain succeed the combined construction and evaluation time, then this preconditioner can be useful for the framework. Thus, we compare FISTA and M_2 preconditioned FISTA for their total time and per iteration time consumption. This is tabulated in Tab. 7.2.

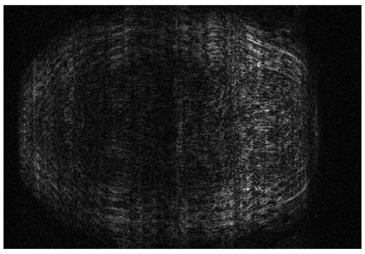
Tab. 7.2 represents the iterations, total time and per iteration time consumed by FISTA and M_2 preconditioned FISTA. Since, both the stopping criteria provides (almost) same results, comparison is made with residual error stopping criteria. It can be observed that M_2 preconditioned FISTA takes half the number of iterations consumed by FISTA at both $r = 2$ and $r = 4$. Comparing the total time consumption of the two algorithms, it can be analysed that M_2 preconditioned FISTA reduces the image reconstruction time by approximately 42% at $r = 2$ and approximately 44% at $r = 4$. At the same time, M_2 preconditioned FISTA increases the per iteration time of the algorithm by 20%. Also, the construction time of M_2 preconditioner was noted which



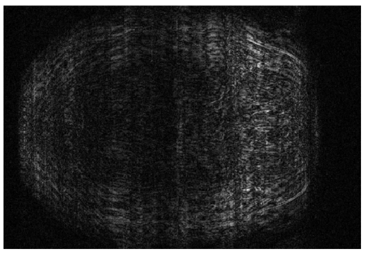
(a)



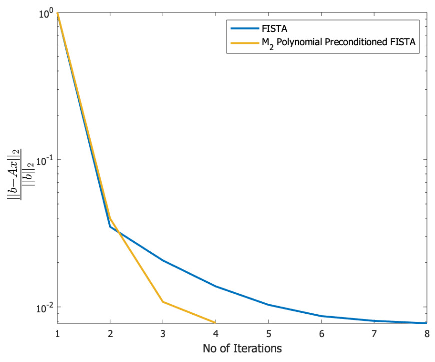
(b)



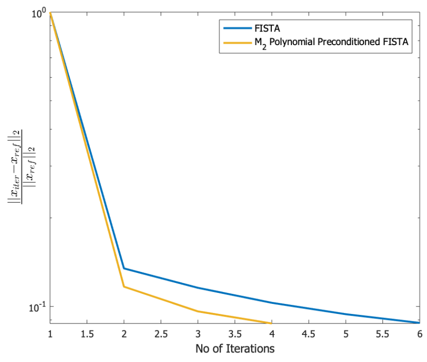
(c)



(d)



(e)



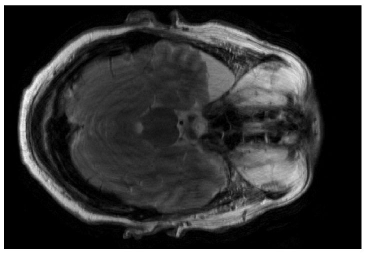
(f)

Figure 7.8: Reconstructed image, difference image and convergence plots for polynomial preconditioned FISTA with $r = 2$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

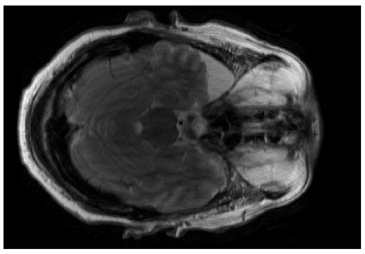
was found to be approximately 0.48 sec. This implies that the construction time of M_2 is equivalent to one additional iteration of the algorithm. Taking the construction time into account, M_2 preconditioned FISTA still consumes 25% lesser time than FISTA at $r = 2$ and 30% lesser time at $r = 4$. Although M_2 preconditioner is computationally expensive, it still manages to provide 25% reduction in overall time (in this case).

As seen before, that block diagonal circulant preconditioned FISTA does not perform well for the current framework, it is insignificant to compare its iteration and time consumption. The block diagonal circulant preconditioner might not have worked for the current framework but it may be promising for other frameworks, for example, preconditioned conjugate gradient algorithm or Split Bregman framework, because the step size of these frameworks do not depend on the eigenvalues of the system matrix. However, there is not enough proof to substantiate the claim. So, it remains an open problem that requires further investigation.

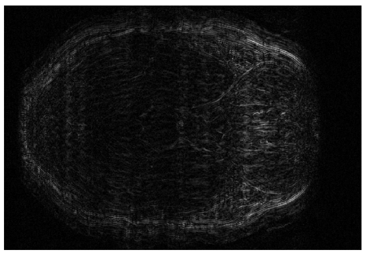
Even then, there is no loss to compare the computation complexity of the derived block diagonal circulant preconditioner. From Tab. 7.1, it can be noted that its construction is approximately one half of the flops consumed by evaluation of A on a vector and its evaluation is approximately 0.04 times the flops consumed by evaluation of A on a vector. Therefore evaluation of this preconditioner on vector is negligible with respect to the flops consumed by A on vector. Further, the construction and evaluation of this preconditioner consumes approximately half the flops consumed by circulant preconditioner. Hence, this preconditioner succeeds all the compared preconditioners in terms of computation complexity.



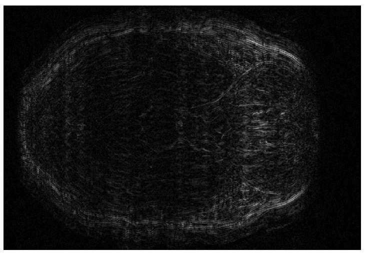
(a)



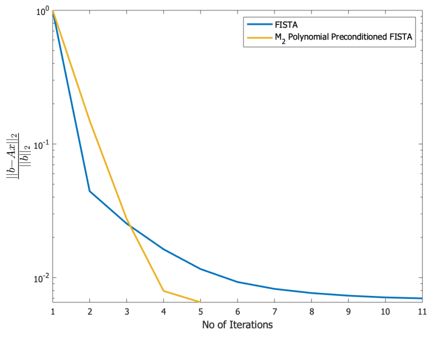
(b)



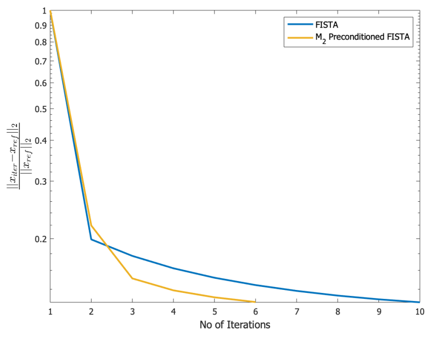
(c)



(d)

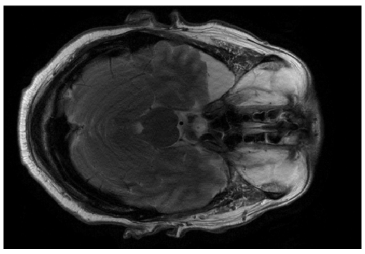


(e)

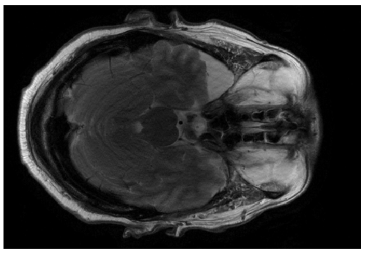


(f)

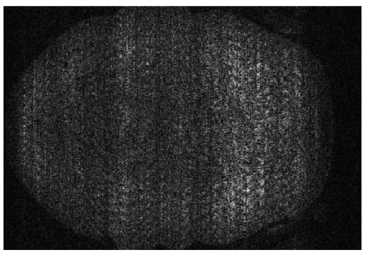
Figure 7.9: Reconstructed image, difference image and convergence plots for Polynomial preconditioned FISTA with $r = 4$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.



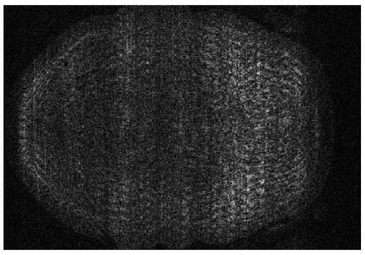
(a)



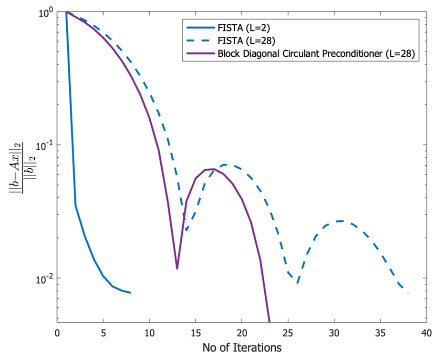
(b)



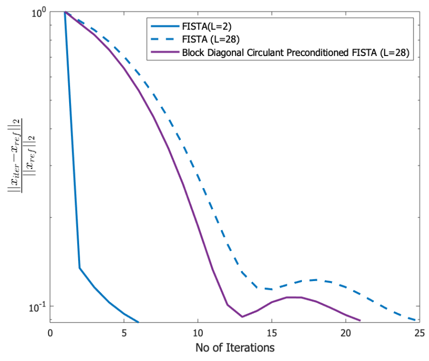
(c)



(d)

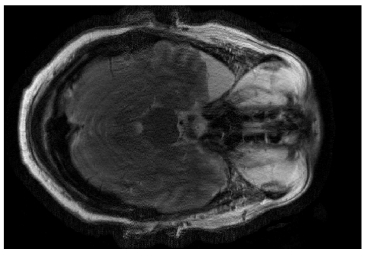


(e)

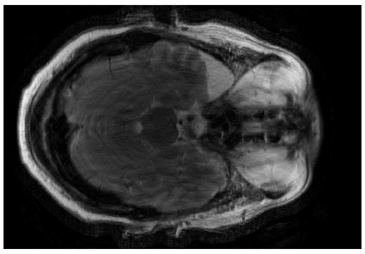


(f)

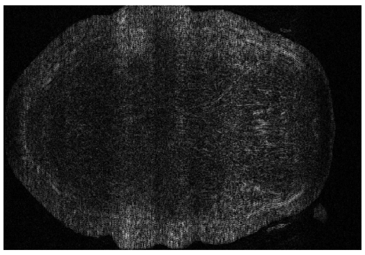
Figure 7.10: Reconstructed image, difference image and convergence plots for block diagonal circulant preconditioned FISTA with $r = 2$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.



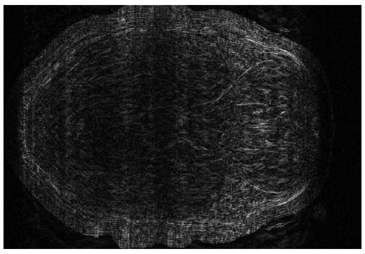
(a)



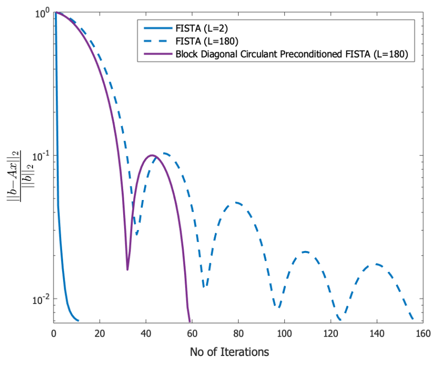
(b)



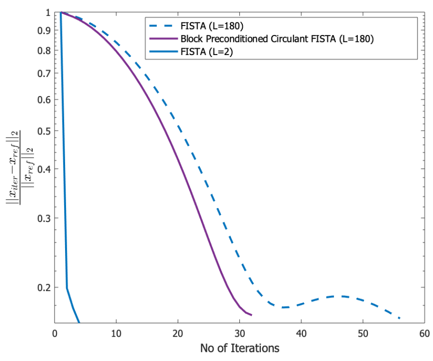
(c)



(d)



(e)



(f)

Figure 7.11: Reconstructed image, difference image and convergence plots for block diagonal circulant preconditioned FISTA with $r = 4$. (a) Reconstructed image with residual error stopping criteria. (b) Reconstructed image with relative error stopping criteria. (c) Difference($\times 3$) image with residual error stopping criteria. (d) Difference($\times 3$) image with relative error stopping criteria. (e) Residual error convergence plot. (f) Relative error convergence plot.

Table 7.1: Calculation of flops for construction of derived preconditioner and evaluation of preconditioners and system matrix on vector.

Operator	Functions	Flops
Evaluation of A on vector	$\sum_{i=1}^{N_c} \mathrm{S}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i \mathbf{x}$	$N_c(4N+2NlogN)-N$
Construction of M_2	$\mathbf{c}_{:,1}^{j} = A\mathbf{m}_{:,1}^{j-1}$	$N_c(4N+2NlogN)-N$
	$\mathbf{g}_{:,1}^{j} = \mathbf{e}_1 - \mathbf{c}_{:,1}^{j}$	N
	$\alpha_j = \frac{(\mathbf{g}_{:,1}^{j})^H A \mathbf{g}_{:,1}^{j}}{(\mathbf{g}_{:,1}^{j})^H AA \mathbf{g}_{:,1}^{j}}$	$2(N_c(4N+2NlogN)-N)+2N$
	$\mathbf{m}_{:,1}^{j} = \mathbf{m}_{:,1}^{j-1} + \alpha_j \mathbf{g}_{:,1}^{j}$	$2N$
	Total	$2[3N_c(4N+2NlogN)+N]$
Construction of M_3	**Total**	$3[3N_c(4N+2NlogN)+N]$
Construction of Block Diagonal Circulant M	$\left(\mathbf{c}_{1;i,j}^{H}\right)^T = \mathrm{F}_n^H \left(\mathbf{s}_{i,j}^{H}\right) \forall i \in \{1,\ldots,\mathrm{N_c}\}$	$N_c Nlogn$
	$\sum_{i=1}^{N_c} \left(\mathbf{c}_{1;i,j}^{H} \circ \mathbf{c}_{1;i,j}^{T}\right)^T$	$2N_cN-N$
	$\mathrm{F}_n^H[\mathrm{F}_n(\ldots) \circ \mathrm{F}_n(\ldots)]$	$N+3Nlogn$
	$\mathbf{k}^{-1}$	N
	Total	$(3+2N_c)Nlogn+N(2N_c+1)$
Evaluation of M_2 on vector	$(\beta_1 I - \beta_2 A)\mathbf{vec}$	$N_c(4N+2NlogN)+N$
Evaluation of M_3 on vector	$(\beta_1 I - \beta_2 A + \beta_3 A^2)\mathbf{vec}$	$3N_c(4N+2NlogN)+2N$
Evaluation of M on vector	$(\mathrm{F}_n^H \operatorname{diag}\{\mathbf{k}_j\}^{-1} \mathrm{F}_n)\mathbf{vec} \forall j \in \{1,\ldots,m\}$	$N+2Nlogn$

Table 7.2: Comparison of derived preconditioners based on the number of iterations, total time and per iteration time.

S.No.	Algorithm	No. of Iterations	Time	Per Iteration Time
		r=2		
1.	FISTA	8	≈2.79 sec	≈0.34 sec
2.	M_2 preconditioned FISTA	4	≈1.61 sec	≈0.41 sec
		r=4		
1.	FISTA	11	≈3.76 sec	≈0.34 sec
2.	M_2 preconditioned FISTA	5	≈2.10 sec	≈0.41 sec

8
Conclusion, Limitations and Future Work

The work investigates the preconditioning techniques to accelerate fast iterative shrinkage threshold algorithm (FISTA) for CS and PI problem. The study derives two new preconditioners, first is a degree one polynomial of the system matrix, A, and second is a block diagonal matrix where each block is a circulant matrix. The simulations for image reconstruction are obtained at two acceleration factors i.e. $r = 2$ and $r = 4$. To evaluate the performance of the preconditioners two metrics are considered: (1) Residual Error; (2) Relative Error. Additionally, the preconditioners are compared on the computation complexity by measuring the number of floating-point operations (flops) consumed by the preconditioners for their construction and evaluation.

Reconstructed image quality and its error convergence depend on the chosen step size and regularization parameters. In the current work, L controls the step size and holds an inverse relationship with step size. According to Beck et. al. [27], L is the Lipschitz constant of the gradient of the data fidelity term and it depends on the maximum eigenvalue of the system matrix, A/A_{reg} (of the current formulation). The maximum eigenvalues of the system matrices are estimated using the power iteration method. Other regularization parameters are set such that they provide high signal to noise ratio and good (perceived) image quality.

First, the performance of two existing preconditioners with FISTA is investigated where the preconditioners utilized are Jacobi and Circulant preconditioner from the work of Koolstra et al. [6]. Sec. 7.2.1 described the reconstructed images and error convergence plots with Jacobi preconditioned FISTA. The results suggested that the largest eigenvalue of the system matrix is higher including the Jacobi preconditioner which reduces the step size of FISTA and makes it slower. Additionally, the diagonal elements of system matrix were found to be constant (Sec. 4.3.1) because of coil sensitivity normalization implying that the Jacobi preconditioner is only scaling the system matrix with a constant value. Scaling the system matrix with a constant value is not an accurate approximation for the inverse of the system matrix. Hence, it can be concluded that Jacobi preconditioner is not a good choice for the current problem. Next, results of circulant preconditioner in Sec. 7.2.2 indicates that implementing this preconditioner with FISTA increases the magnitude of the largest eigenvalue of the system matrix substantially. Accordingly, the step size is reduced which makes FISTA considerably slow. Although, circulant preconditioner is computationally efficient, yet incorporating it in the current framework does not provide the desired results.

Reconstructed results with polynomial preconditioned FISTA in Sec. 7.2.3 indicate that the convergence with preconditioner is a lower bound to the convergence without preconditioner. Under both evaluation metrics, polynomial preconditioned FISTA takes fewer iterations than FISTA. Comparison of consumed iterations, total time and per iteration time indicates that M_2 preconditioner takes approximately half the number iterations consumed by FISTA. However, M_2 preconditioner increases the per iteration time of FISTA by 20%. Additionally, Sec. 7.3 indicates that M_2 preconditioner is expensive to construct. Therefore, even though M_2 preconditioned FISTA reduces the total time by a factor of 0.25, it is an expensive preconditioner especially when the the number of receiver coils are more.

The results of block-diagonal circulant preconditioned FISTA in Sec. 7.2.4 are similar to the circulant preconditioned FISTA. Analogous to circulant preconditioner, block-diagonal circulant preconditioner also significantly increases the magnitude of the largest eigenvalue of the system matrix resulting in slower FISTA. Sec. 7.3 describes that this preconditioner is computationally cheap to construct and evaluate, even cheaper than the circulant preconditioner. However, for the current framework including this preconditioner comes with the expense of larger maximum eigenvalue. If the step size was independent of the largest eigenvalue of the system matrix, both circulant and block diagonal circulant preconditioner would be the obvious choice.

8.1. Limitations

The major limitation of this work is that both derived preconditioners estimate the inverse of A and not A_{reg}. Therefore, the obtained preconditioners do not exactly approximate the inverse. Better inverse approximation of system matrix can be estimated by taking into account the diagonal matrix $\mathrm{Q}^H\mathrm{Q}$. Thus the problem needs further investigation. Secondly, every time a preconditioner is included in the framework, the step size has to be varied according to the system matrix's largest eigenvalue. This limits us to use computationally efficient preconditioners like circulant and block diagonal circulant. One way to overcome this is by using FISTA with backtracking rule, however, FISTA with backtrack comes with an expense of increased per iteration time of the algorithm.

8.2. Future Work

To yield better results inverse of $\mathrm{Q}^H\mathrm{Q}$ term must be taken in to account when approximating the inverse. While deriving the polynomial preconditioner, it is observed that diagonal of A is constant and hence its performance is based on the fact that the diagonal of the system matrix is constant. Adding the term $\mathrm{Q}^H\mathrm{Q}$ to A refutes the fact that diagonal values are constant. Further investigation can be done to find a preconditioner that exactly approximates the inverse of A_{reg}.

The derived block diagonal circulant preconditioner does not provide the desired results with the FISTA framework, however, its cheap construction and evaluation on a vector indicates that it can be a potential preconditioner for a different framework. Since, the preconditioner is (to great extend) similar to the circulant preconditioner of [6], block diagonal circulant preconditioner can be tried with Split Bregman framework.

The study tells us that to include preconditioners with FISTA framework, the largest eigenvalue of the system matrix must not change. Therefore, further investigation is required to find preconditioners that without changing the largest eigenvalue, improves the condition number of the system matrix and simultaneously are cheap to construct and evaluate.

A

NOMENCLATURE

The report follows the nomenclature given in Tab. A.1 unless otherwise stated in the document.

Table A.1: Nomenclature of the document

Quantity	**Notation**
Scalar	a
Vector	$\mathbf{a}$
Vector transpose	$\mathbf{a}^T$
Vector hermitian	$\mathbf{a}^H$
i^{th} component of vector	a_i
Matrix	A
Matrix transpose	A^T
Matrix hermitian	A^H
Entry of matrix A on i^{th} row and j^{th} column	a_{ij}
i^{th} row of matrix A	$\mathbf{a}_{i:}$
j^{th} column of matrix A	$\mathbf{a}_{:j}$
Set of $m \times n$ matrix with real entries	$\mathbb{R}^{m\times n}$
Set of $m \times n$ matrix with complex entries	$\mathbb{C}^{m\times n}$
Set of real column vectors with n components	$\mathbb{R}^n$ and $\mathbb{R}^{n\times 1}$
Set of complex column vectors with n components	$\mathbb{C}^n$ and $\mathbb{C}^{n\times 1}$
ℓ_2 norm	$\|\cdot\|_2$
ℓ_1 norm	$\lvert\cdot\rvert_1$
Absolute function	$\lvert\cdot\rvert$
Signum function	sgn
$\max(a,0)$, $a \in \mathbb{R}$	$(a)_+$
Hadamard product	$\circ$
Convolution operator	$*$
Assignment operator	$:=$

B

Proof of Shrinkage Operator

To prove the property defined in Eq. B.11, we reform the compressed sensing problem as function of **c**, where **c** = W**x**, instead of **x** described by the equations below

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathrm{N_c}} \left\| \mathrm{RFS}_i \mathrm{W}^H \underbrace{\mathrm{W}\mathbf{x}}_{\mathbf{c}} - \mathbf{m}_i \right\|_2^2 + \lambda | \underbrace{\mathrm{W}\mathbf{x}}_{\mathbf{c}} |_1 \right\} \tag{B.1}$$

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{\mathrm{N_c}} \left\| \mathrm{RFS}_i \mathrm{W}^H \mathbf{c} - \mathbf{m}_i \right\|_2^2 + \lambda |\mathbf{c}|_1 \right\}. \tag{B.2}$$

where, $\mathrm{W}^H\mathrm{W} = I$. Accordingly, the functions f and g are now expressed as $f(\mathbf{c}) = \sum_{i=1}^{\mathrm{N_c}} \left\| \mathrm{RFS}_i \mathrm{W}^H \mathbf{c} - \mathbf{m}_i \right\|_2^2$ and $g(\mathbf{c}) = \lambda|\mathbf{c}|_1$. Taking the derivative of the following equation

$$p_L(\mathbf{w}) = \underset{\mathbf{c}}{\operatorname{argmin}} \left\{ g(\mathbf{c}) + \frac{L}{2} \left\| \mathbf{c} - \left(\mathbf{w} - \frac{1}{L} \nabla f(\mathbf{w}) \right) \right\|_2^2 \right\} \tag{B.3}$$

results in

$$\frac{\partial p_L(\mathbf{w})}{\partial \mathbf{c}} = \lambda \operatorname{sgn}(\mathbf{c}) + \frac{2L}{2} \left[\mathbf{c} - \left(\mathbf{w} - \frac{1}{L} \nabla f(\mathbf{w}) \right) \right]. \tag{B.4}$$

Equating the derivative to zero and replacing **w** with $\mathbf{c}_{k-1}$ results in

$$\mathbf{c} + \frac{\lambda}{L} \operatorname{sign}(\mathbf{c}) = \mathbf{c}_{k-1} - \frac{1}{L} \nabla f(\mathbf{c}_{k-1}). \tag{B.5}$$

Now, solving this equation for **c** yields the following solution

$$\mathbf{c} = \begin{cases} \mathbf{v} + \frac{\lambda}{L} & \mathbf{v} < \frac{\lambda}{L} \\ 0 & |\mathbf{v}| < \frac{\lambda}{L} \\ \mathbf{v} - \frac{\lambda}{L} & \mathbf{v} > \frac{\lambda}{L} \end{cases} \tag{B.6}$$

where, $\mathbf{v} = \mathbf{c}_{k-1} - \frac{1}{L} \nabla f(\mathbf{c}_{k-1})$. Alternatively, Eq. B.6 can efficiently be written in the following form

$$\mathbf{c}_k = \mathcal{T}_\alpha \left(\mathbf{c}_{k-1} - \frac{1}{L} \nabla f(\mathbf{c}_{k-1}) \right) \tag{B.7}$$

where $\alpha = \frac{\lambda}{L}$ and $\mathcal{T}_\alpha$ is the shrinkage operator defined as

$$\mathcal{T}_\alpha(\mathbf{v}) = (|\mathbf{v}| - \alpha)_+ \operatorname{sgn}(\mathbf{v}). \tag{B.8}$$

Here, the gradient $\nabla f(\mathbf{c})$ is given by

$$\nabla f(\mathbf{x}) = 2 \sum_{i=1}^{N_c} \mathrm{WS}_i^H \mathrm{F}^H \mathrm{R}^H \mathrm{RFS}_i \mathrm{W}^H \mathbf{c} - 2 \sum_{i=1}^{N_c} \mathrm{WS}_i^H \mathrm{F}^H \mathrm{R}^H \mathbf{m}_i. \tag{B.9}$$

To obtain the solution in terms of **x**, substitute the value of **c** in Eq. B.7. The substitution results in

$$\begin{aligned} \mathrm{W}\mathbf{x}_k &= \mathscr{T}_\alpha \left(\mathrm{W} \left(\mathbf{x}_{k-1} - \frac{1}{L} \nabla f\left(\mathbf{x}_{k-1}\right) \right) \right) \\ \mathbf{x}_k &= \mathrm{W}^H \mathscr{T}_\alpha \left(\mathrm{W} \left(\mathbf{x}_{k-1} - \frac{1}{L} \nabla f\left(\mathbf{x}_{k-1}\right) \right) \right) \end{aligned} \tag{B.10}$$

Eq. B.10 proves that if $g(\mathbf{x}$ is a function of an operator such as $|\Psi(\mathbf{x})|_1$ where Ψ is unitary sparsifying transform operator (i.e. $\Psi^H \Psi = I$), then the shrinkage operator is defined as

$$\mathscr{T}_\alpha(\mathbf{x}) = \Psi^H \mathscr{T}_\alpha(\Psi(\mathbf{x})). \tag{B.11}$$

Bibliography

[1] I. I. Rabi, J. R. Zacharias, S. Millman, and P. Kusch, *A new method of measuring nuclear magnetic moment,* Phys. Rev. **53**, 318 (1938).

[2] P. C. Lauterbur, *Image formation by induced local interactions: examples employing nuclear magnetic resonance,* nature **242**, 190 (1973).

[3] B. P. Sutton, D. C. Noll, and J. A. Fessler, *Fast, iterative image reconstruction for mri in the presence of field inhomogeneities,* IEEE transactions on medical imaging **22**, 178 (2003).

[4] S. Ramani and J. A. Fessler, *An accelerated iterative reweighted least squares algorithm for compressed sensing mri,* in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (IEEE, 2010) pp. 257–260.

[5] M. J. Muckley, D. C. Noll, and J. A. Fessler, *Fast, iterative subsampled spiral reconstruction via circulant majorizers,* (2016).

[6] K. Koolstra, J. Gemert, P. Börnert, A. Webb, and R. Remis, *Accelerating compressed sensing in parallel imaging reconstructions using an efficient circulant preconditioner for cartesian trajectories,* Magnetic Resonance in Medicine **81** (2018), 10.1002/mrm.27371.

[7] M. A. Griswold, P. M. Jakob, M. Nittka, J. W. Goldfarb, and A. Haase, *Partially parallel imaging with localized sensitivities (pils),* Magnetic Resonance in Medicine **44**, 602 (2000).

[8] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger, *Sense: sensitivity encoding for fast mri,* Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine **42**, 952 (1999).

[9] M. A. Griswold, P. M. Jakob, R. M. Heidemann, M. Nittka, V. Jellus, J. Wang, B. Kiefer, and A. Haase, *Generalized autocalibrating partially parallel acquisitions (grappa),* Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine **47**, 1202 (2002).

[10] D. K. Sodickson and W. J. Manning, *Simultaneous acquisition of spatial harmonics (smash): fast imaging with radiofrequency coil arrays,* Magnetic resonance in medicine **38**, 591 (1997).

[11] M. Lustig and J. M. Pauly, *Spirit: iterative self-consistent parallel imaging reconstruction from arbitrary k-space,* Magnetic resonance in medicine **64**, 457 (2010).

[12] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, *Espirit—an eigenvalue approach to autocalibrating parallel mri: where sense meets grappa,* Magnetic resonance in medicine **71**, 990 (2014).

[13] D. L. Donoho, *Compressed sensing,* IEEE Transactions on information theory **52**, 1289 (2006).

[14] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,* IEEE Transactions on information theory **52**, 489 (2006).

[15] M. Rabbani, *Jpeg2000: Image compression fundamentals, standards and practice,* Journal of Electronic Imaging **11**, 286 (2002).

[16] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, *Compressed sensing mri,* IEEE signal processing magazine **25**, 72 (2008).

[17] H. H. Bauschke, P. L. Combettes, *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces,* Vol. 408 (Springer, 2011).

[18] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*, Vol. 15 (Springer Science & Business Media, 2003).

[19] Y. Saad, *Iterative methods for sparse linear systems* (SIAM, 2003).

[20] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,* Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences **57**, 1413 (2004).

[21] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing,* in *Fixed-point algorithms for inverse problems in science and engineering* (Springer, 2011) pp. 185–212.

[22] M. A. Figueiredo and R. D. Nowak, *An em algorithm for wavelet-based image restoration,* IEEE Transactions on Image Processing **12**, 906 (2003).

[23] M. A. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak, *Majorization–minimization algorithms for wavelet-based image restoration,* IEEE Transactions on Image processing **16**, 2980 (2007).

[24] T. Goldstein and S. Osher, *The split bregman method for l1-regularized problems,* SIAM journal on imaging sciences **2**, 323 (2009).

[25] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction,* SIAM Journal on Imaging Sciences **1**, 248 (2008).

[26] J. M. Bioucas-Dias and M. A. Figueiredo, *A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration,* IEEE Transactions on Image processing **16**, 2992 (2007).

[27] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems,* SIAM journal on imaging sciences **2**, 183 (2009).

[28] Y. Nesterov, *Gradient methods for minimizing composite functions,* Mathematical Programming **140**, 125 (2013).

[29] M. Elad, B. Matalon, and M. Zibulevsky, *Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization,* Applied and Computational Harmonic Analysis **23**, 346 (2007).

[30] K. T. Block, M. Uecker, and J. Frahm, *Undersampled radial mri with multiple coils. iterative image reconstruction using a total variation constraint,* Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine **57**, 1086 (2007).

[31] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging,* Journal of mathematical imaging and vision **40**, 120 (2011).

[32] M. J. Muckley, D. C. Noll, and J. A. Fessler, *Fast parallel mr image reconstruction via b1-based, adaptive restart, iterative soft thresholding algorithms (barista),* IEEE transactions on medical imaging **34**, 578 (2014).

[33] A. J. Wathen, *Preconditioning,* Acta Numerica **24** (2015).

[34] N. H. Clinthorne, T. . Pan, P. . Chiao, W. L. Rogers, and J. A. Stamos, *Preconditioning methods for improved convergence rates in iterative reconstructions,* IEEE Transactions on Medical Imaging **12**, 78 (1993).

[35] S. Ramani and J. A. Fessler, *Parallel mr image reconstruction using augmented lagrangian methods,* IEEE Transactions on Medical Imaging **30**, 694 (2010).

[36] A. E. Yagle, *New fast preconditioners for toeplitz-like linear systems,* in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing,* Vol. 2 (2002) pp. II–1365–II–1368.