Titre: Title:	Fast Iterative Reconstruction in X-Ray Tomography Using Polar Coordinates
Auteur: Author:	Mahsa Aliakbar Golkar
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Aliakbar Golkar, M. (2013). Fast Iterative Reconstruction in X-Ray Tomography Using Polar Coordinates [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1204/

UNIVERSITÉ DE MONTRÉAL

FAST ITERATIVE RECONSTRUCTION IN X-RAY TOMOGRAPHY USING POLAR COORDINATES

MAHSA ALIAKBAR GOLKAR
INSTITUT DE GÉNIE BIOMÉDICAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE BIOMÉDICAL)
AÔUT 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

FAST ITERATIVE RECONSTRUCTION IN X-RAY TOMOGRAPHY USING POLAR COORDINATES

présenté par : ALIAKBAR GOLKAR Mahsa
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. SAVARD Pierre, Ph.D., président
M. GOUSSARD Yves, Ph.D., membre et directeur de recherche
Mme CHERIET Farida, Ph.D., membre

This thesis is dedicated to my parents.
For their endless love, support and encouragement . . .

ACKNOWLEDGMENT

During my studies in École polytechnique de Montréal, I had a rich academic, intellectual and social experience. This inspired me to work even harder to reach higher academic standards for which I would like to thank several people.

First, I would like to thank my supervisor, Professor Yves Goussard for his guidance and valuable comments in the development of this thesis. My appreciation also goes to my colleagues at École polytechnique de Montréal for creating a pleasant atmosphere to work in. In particular I would like to thank our research group: Adrien Wagner, Matthieu Voorons, Corentin Friedrich and Ju Jing Tan for making office an inspiring and fun workplace.

My gratitude extends to my family and friends who supported me during my studies. First and foremost, I would like to thank my dear parents who have always been present throughout my life in spite of the distance. I am thankful of them for always believing in me and supporting me in my decisions, without whom I could not have made it here. I owe special thanks to my lovely sister and brother, Noushin and Alireza for their support and for motivating me to go after higher achievements. I could not ask for a better sister and brother.

Last, but not least, I would like to acknowledge my friends in Montréal whose friendship means a lot to me. In particular, I would like to thank my supportive friend, Arash Moradi, for his moral support and constant encouragement. His optimistic view on things always made me to carry on with confidence. My special gratitude to two of my best friends Mina Ranjbaran and Kian Jalaleddini for being with me in thicks and thins of life in the past five years. I find myself lucky to have friends like them in my life.

RÉSUMÉ

Notre objectif est de réduire l'espace mémoire nécessaire ainsi que le temps de reconstruction excessif des méthodes itératives en imagerie à rayons X. En général, les méthodes itératives permettent d'obtenir une meilleure qualité de reconstruction que celles obtenues par FBP. Cela est dû à l'utilisation d'un modèle plus précis dans le processus de reconstruction. En effet, le modèle prend en compte le bruit et peut introduire un certain a priori sur l'image à reconstruire. Le problème peut alors être résolu par des techniques d'optimisation.

En reconstruction d'images par méthodes itératives, la taille importante de la matrice de projection joue un rôle prédominant dans la mémoire requise par ce type de méthodes. Le temps de reconstruction, quant à lui, est rallongé par le grand nombre d'opérations de projection et rétroprojection. Ces aspects nécessitent une attention particulière lors de reconstructions par approches itératives.

L'objectif de cette maîtrise a été de s'attaquer à ces deux aspects en développant une technique efficace de reconstruction d'images médicales. L'hypothèse de rayons X monochromatiques est utilisée et l'invariance en coordonnées polaires des tomographes commerciaux est considérée. En effet, l'utilisation de coordonnées polaire pour représenter l'objet permet d'obtenir une certaine redondance dans les coefficients de la matrice de projection. Celle-ci est parcimonieuse et a une structure bloc-circulante, ce qui mène à une réduction significative de l'espace mémoire nécessaire pour la stocker. Mais ce type de représentation entraîne des questions de qualité de reconstruction ainsi que des questions sur les aspects numériques des méthodes. Ce travail aborde les problèmes soulevés.

Comme mentionné plus haut, le temps de reconstruction en tomographie rayons X est essentiellement déterminé par le temps de calcul des opérations de projection et rétroprojection qui sont réalisées à chaque itération. La parallélisation des calculs permet de réduire le temps de reconstruction de manière significative, ceci est abordé ici. De plus, la conception de préconditionneurs adaptés à la fonction objectif entraîne une amélioration de la vitesse de convergence des méthodes itératives.

Ce travail consiste en une étude préliminaire sur les performances de reconstruction d'images tomographiques en se basant sur une représentation polaire des objets à imager. Les résultats obtenues dans ce mémoire peuvent être utilisés pour la reconstruction d'images cliniques 3D.

Il est aussi possible d'étendre les algorithmes développés ici à un modèle polychromatique et aussi de réduire les artefacts métalliques.

ABSTRACT

We aim at reducing the high memory need and the long reconstruction time of the iterative methods for reconstructing the X-ray tomography images. In general, iterative methods are capable of providing a higher quality reconstructed image compared to those obtained through filtered backprojection. This is because in the iterative methods, a more accurate model is used in the reconstruction process. The model used in this technique accounts for the noise and can incorporate some prior knowledge on the image and therefore can provide images with higher quality compared to those obtained using the filtered backprojection technique. The reconstruction problem can then be solved using optimization techniques.

In using iterative methods for image reconstruction, the large size of the projection matrix is the main cause of having high memory need in this method. Moreover, requiring to perform projection and backprojection operations numerous times is the main reason for the long reconstruction time. These problems need to be addressed properly for wider adoption of the iterative approaches in image reconstuction.

The work presented herein aims at addressing these problems by developing an efficient technique which makes reconstruction of clinical size images possible. This will be done in a simple framework under the assumption of a monochromatic X-ray source. The objective is fulfilled by considering the fact that the geometry of commercial tomographs is invariant in polar coordinates. Using polar coordinates for representing the object, the coefficients of the projection matrix will be highly redundant. The matrix is also very sparse and has a block-circulant structure. Consequently, using polar coordinates for representing the object leads to a significant decrease in memory requirement. There are some questions associated with this type of representation which include numerical efficiency of the reconstruction process using this type of representation and actual quality of reconstructed image. This work tries to study and address these questions.

As already mentioned, reconstruction time of tomography problems is mainly determined by the computation time of projection and backprojection operations that need to be performed at each iteration. The parallel implementation of these operations can reduce the reconstruction time significantly and is addressed here. Moreover, by designing preconditioners tailored to the structure of the objective function a sufficient increase in the convergence speed of iterative methods was achieved.

The current work is a preliminary study on the efficiency of using polar coordinates for representing the object and reconstructing the tomography images. The results which have been obtained in this work can now be used for developing the 3D reconstruction of clinical data.

We can also use the developed algorithms in this work to expand the current framework to

polychromatic model and benefit from the efficiency of this model in reducing the metal artifacts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

CT	Computed Tomography
PET	Positron Emission Tomography
SPECT	Single-Photon Emission Computed Tomography
FBP	Filtered Back-Projection
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
GGMRF	Generalized Gaussian Markov random Field
CG	Conjugate gradient
GA	Gradient Ascent
ICD	Iterative Coordinate Descent
MAP	Maximum A Posteriori
ML	Maximum Likelihood
SPS	Separable Paraboloidal Surrogates
OS	Ordered Subset
OS-SPS	Ordered Subset - Separable Paraboloidal Surrogates
PR-NLCG	Polak-Ribiere Nonlinear Conjugate gradient
TRIOT	Transmission Incremental Optimization Transfer
FR	Fletcher-Reeves
PR	Polak-Ribière
HS	Hestenes-Stiefel
SNR	Signal to Noise ratio
L-BFGS-B	Limited-memory Broyden-Fletcher-Goldfarb-Shanno with bound constraint
BC	Block-Circulant
RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

Medical imaging is a technique used to create images of the human body for medical purposes; they make accurate diagnosis possible. There are many benefits associated with it, e.g., more evidence-based decision making and better understanding of the effect of treatments on diseases. X-ray Computed Tomography (CT) is one of the most widely used techniques for reconstructing medical and industrial images. In this technique, an object is placed between a rotating source and an array of detectors and its attenuation characteristic is obtained by transmitting a collection of photons along different angles and measuring the number of unabsorbed photons reaching the detectors (Fig.(1.1)). The data collected by detectors are called projection data and are used to reconstruct a (cross) section of the human body.

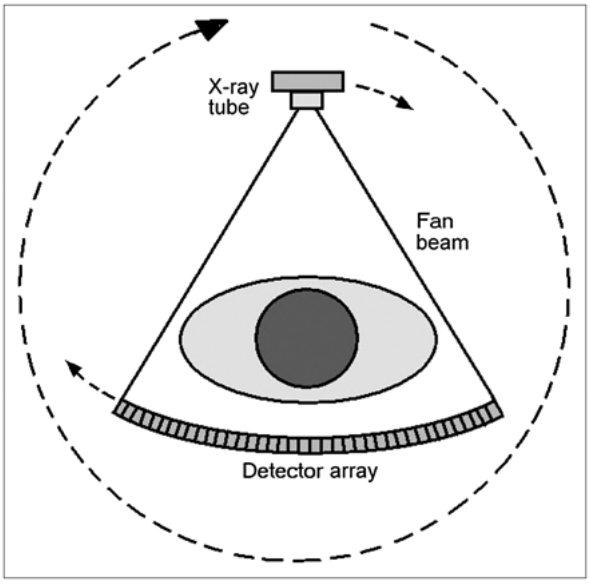


Figure 1.1 Data Acquisition in computed tomography ©(Goldman, 2007).

In general CT is used for diagnosis purposes. Some well-known examples of wide spread use of CT are: for detecting tumors and hemorrhage in head, imaging of the coronary arteries in cardiac CT angiography and, diagnosis of abdominal diseases and cancer, to name a few. Another recent use of CT is in surgeries which require accurate intra-operative imaging. This is more evident for surgeries, which require more safety and need to be less invasive. Some examples are spinal, cardiac, vascular and orthopedic surgeries (Fayyazi *et al.*, 2004). In such surgeries, the

reconstruction time is of great importance in addition to the quality of reconstructed images.

Consequently, for imaging the vital organs where precise images are required for an accurate diagnosis, having a precise, noise-free and artifact-free image in a decent amount of time is necessary.

1.1 Image reconstruction in X-ray tomography

In general, the methods which are used for reconstructing the object can be divided into two groups: analytical and iterative methods. Current CT scanners usually use the analytical methods for reconstructing the object. This is because this technique is fast and provides images in a short amount of time. The numerical efficiency of this technique is mainly due to the simplifications which have been made in the model used for reconstruction. Noise is not modeled in this technique and therefore for highly noisy measurements or low-dose experiments, the quality of the reconstructed image may suffer from noise. It has been attempted to reduce the noise level using post-processing techniques, however, further improvement requires a more complicated model which accounts for noise, and benefits from estimation techniques.

Another well-known situation where the quality of analytical reconstructions may become unsatisfactory is in the presence of metal objects. This is because the metal objects behave very differently from the other soft tissues in human body and they will result in significant reduction in the number of photons reaching detectors. Consequently, treating them in a same way as soft tissues will result in having special type of artifacts known as metal artifacts. An example of such cases is in cardiac imaging where metal stents are placed in the arteries or in orthopedic surgeries where large metal objects are present (Rao *et al.*, 2003; Santos *et al.*, 2011). A low quality reconstruction may make the diagnosis impossible or less accurate in some cases. To overcome this problem one needs to use a more accurate model and solve the problem using *iterative methods*.

Iterative methods, are found efficient in improving the quality of images. These methods, consider the noise model and allow incorporating some prior knowledge on image. Moreover, this technique has the capability to model the energy-dependency of the beams and the potential to be used for different scanning geometries. As a result, they have proved to outperform analytical methods in CT (De Man *et al.*, 2000; Wang *et al.*, 1999).

Another interesting advantage of the iterative methods as opposed to the analytical methods is the relationship between noise and X-ray dose. For any given quality of measurements, iterative methods provide a higher quality reconstruction compared to the analytical methods in terms of trade-off between resolution and noise. This is mainly due to the use of a more accurate model in this method by considering the noise model explicitly. For many applications, quality of the analytical reconstruction is satisfactory however, it is becoming more important to have a satisfac-

tory reconstruction with a lower radiation dose. A lower radiation dose is equivalent to a lower measurement quality and a lower signal-to-noise ratio for the measurement data. Therefore, using iterative methods one can obtain same quality of reconstruction as with the analytical method with higher dose, with a reduced radiation dose.

In this technique, an image is reconstructed by minimizing a cost function, iteratively. Two operations called projection and backprojection are performed at every iteration, which are the core of process and account for the main time of reconstruction. These operations will be discussed in detail in subsequent chapters. The main drawbacks of this technique is known to be its high memory need and long reconstruction time. This problem is more challenging in X-ray CT imaging compared to PET and SPECT due to the higher volume of data involved. The problem becomes even more complicated for 3D reconstruction and consequently, hinder the use of iterative methods in practice.

1.2 Objectives

The long-term goal of our research group is to develop efficient techniques to reconstruct tomography images using iterative methods, and at the same time try to preserve the low memory need and reconstruction time of analytical methods, as much as possible. Having a more accurate model when iterative methods are used allows improving the quality of image and decreasing the noise and the artifacts. These techniques will be developed such that efficient implementation of projection and backprojection operations will be made possible, leading to reduction of the reconstruction time. The results will then be applicable to all imaging techniques with similar data acquisition geometry or similar type of calculations. Considering the fact that projection and backprojection are the main elements in all imaging techniques, results on efficient implementation of these operations can be useful for SPECT and PET imaging as well.

As already discussed, the main bottleneck of iterative methods which hinders their use in practical cases is known to be their long reconstruction time. Different efforts have been made to address this problem. These efforts essentially lie in three categories: algorithmic improvement, dedicated hardware and parallel processing.

In this work, I will first attempt to reduce the high memory need of iterative methods by using a data formation which takes advantage of the maximum symmetry in the system. To do this, cylindrical coordinates will be used for representing the object. I will then try to develop an efficient optimization algorithm with adequate computation time and memory requirement, which can be used to reconstruct images in a decent amount of time.

To summarize, the general objective of this work can be divided into three sub-goals:

1. Derivation of a model consistent with the cylindrical discretization scheme.

2. Acceleration of the optimization algorithm used for reconstructing the image.

3. Validation of results by using numerical phantoms.

1.3 Thesis structure

This thesis is is organized as follows:

In *Chapter 2*, the basic principles of X-ray tomography will be discussed and the linear forward model which will be used for reconstruction is provided. We will then discuss the reconstruction problem briefly and will study the types of estimations available and the parameters involved for solving ill-posed problems. Some generally-used numerical methods for image reconstruction will then be reviewed and compared. This will be followed by a short review on the effect of conditioning[1] on speed of convergence and use of preconditioners[2]. We will finish this chapter by studying different types of object discretization.

In *Chapter 3*, the general framework of this dissertation will be provided and using the discussion of Chapter 2, the type data formation for discretizing the object and the numerical method which will be used for reconstructing the image will be provided. A summary of the objectives which will be followed in Chapter 4, 5 and 6 will then be given.

In *Chapter 4*, discretization of the object in cylindrical coordinates will be studied. This includes choosing the tessellation scheme for discretization of the object and deriving a mathematical model consistent with this type of representation. We will then derive the algorithm for efficient implementation of projection and backprojection. The effect of this representation on the convergence of the problem when numerical method are used will then be discussed and the requirement of designing an appropriate preconditioner will be justified. This will fulfill the first objective of this work.

In *Chapter 5*, the structure of normal matrix which has a strong impact on the convergence speed of the numerical methods for solving the problem, will be studied and possibility of designing different type of preconditioners will be explored. Two approaches will be followed for designing preconditioners: block-diagonal approximation to normal matrix and block-diagonalization of normal matrix using fast Fourier transform (FFT). For each case, two type of preconditioners will be developed: diagonal and incomplete Cholesky. The limitation and the anticipated problems for each approach will be explored. This accomplish the second objective of this work.

In *Chapter 6*, simulations results are provided. We will first compare the performance of two

[1] Conditioning of a problem is defined by how different the largest and smallest eigenvalues are, and how wide the eigenvalues are spread.

[2] Preconditioners improve the convergence of numerical methods by transforming the problem to one which is easier to solve (will be discussed in detail in Chapter 5).

of penalty function[1] discussed in Chapter 4 for polar-coordinates. Efficiency of preconditioners developed in Chapter 5 will then be studied and the quality of reconstructed image using different penalty function will be evaluated. A conclusion will be made on the effectiveness of each method. This will fulfill the third objective of this work.

In *Chapter 7*, we conclude by drawing some conclusions on the present work and some suggestions for the future works.

[1] In solving ill-posed problems, a penalty function is usually added to incorporate some prior knowledge on problem and ensures that a unique solution always exists (this will be discussed in detail in Chapter 3)

CHAPTER 2

STATE OF THE ART

In this work we are interested in the images of CT scanners with the axial model for data acquisition. These scanners provide a 3D image of a volume of the object. This chapter provides a short study on the X-ray tomography and the model used in this technique. The available methods used for image reconstruction in this technique is also provided and the advantages and disadvantages of each method is discussed.

2.1 X-ray Tomography

In Computed tomography (CT), a large number of X-rays are transmitted through the object (or human anatomy) for producing the image. The X-rays are produced in a region which is nearly a point source and then are transmitted through the object. These rays are then either absorbed by the anatomy to be imaged or are scattered through the object and collected by the detectors. This process is repeated for all projections angles with source and detectors rotating around the object to be imaged. The data collected by detectors are called projection data or sinogram. The number of unabsorbed photons reaching the detectors will then be used to reconstruct the image (Mudry *et al*., 2003).

In practice, real medical objects, i.e., patients, are three dimensional. Therefore, a volumetric representation of the anatomy to be imaged and 3D image reconstruction is of great interest. In general, data acquisition for 3D CT imaging can be performed by two different types of scanners, axial and helical (or spiral). In axial scanners, a cone-beam geometry is used and a number of slices are reconstructed from patients body when he/she is lying on a steady bed. To acquire more images for volumetric representation of object, this process needs to be repeated with the bed/patient repositioned. In spiral scanners, the patient is translated continuously through the gantry at a specific speed and multiple projection datasets are acquired. This makes the data acquisition time to be reduced. However, the reconstruction of images using this type of data acquisition is more complicated as one needs to accommodate the spiral path traced by the X-ray source.

2.2 Models for X-ray tomography

In transmission tomography, the goal is to obtain the spatial distribution of the linear attenuation coefficient of discretized object $\mu(\vec{x},\mathscr{E})$, where $\vec{x}=(x_1,x_2,x_3)$ denote the spatial location in 3-space

and $\mathscr{E}$ represents the beam energy. The units of μ are typically inverse centimeters (cm^{-1}).

2.2.1 Transmission measurements

In X-ray CT, projection data is made up of a set of line integrals from different projection angles. The line integral of each projection angle is the sum of the attenuation coefficient of an absorptive material which the X-ray beam is traveling through in a straight line.

X-ray sources in CT scanners usually emit photons with a continues energy distribution spectrum and are known to be polychromatic. In medical X-ray imaging this energy is typically between 5 and 150 keV, and the energy level is adjusted based on the anatomy of interest and purpose of imaging (Mudry *et al.*, 2003). Characterizing the spatial location of object in 3-space using X-ray linear attenuation coefficient $\mu(x_1,x_2,x_3)$, the projection measurement of each ray y_i (mean number of photons), recorded by the i-th detector follows the relationship below (Elbakri et Fessler, 2002):

$$y_i = \int I_0(\mathscr{E}) exp\{-\int_{L_i} \mu(x_1,x_2,x_3,\mathscr{E})dl\}d\mathscr{E} \tag{2.1}$$

where I is the beam intensity, μ represents the linear attenuation of the object and L_i denote the trajectory of the ray through the object.

In general considering the polychromatic nature of the beam, photons passing through different materials in the object are absorbed or transferred differently. Photons with lower energy have a greater chance to be absorbed by the object, compared to those with higher energy level. Consequently, the mean energy of beam increases gradually. This is known as the beam-hardening phenomenon and not accounting for this phenomenon in the transmission model will generate artifacts in the reconstructions. These artifacts are especially evident in the presence of the metal objects in the anatomy to be imaged.

Many researchers neglect the polychromatic nature of the X-ray source and use the monochromatic model for simplicity. The reason is that when imaging the human anatomy, the soft tissues usually behave similarly in different energy levels and so this assumption is valid and does not introduce large inaccuracy in the model. In using monochromatic model, we make the assumption that the X-ray attenuation coefficients are independent of the X-ray source energy, and thus, the "Beer-Lambert law" can be used. The intensity of the ray measured as it leaves the object then follows: $I = I_0 exp\{-\mu L\}$, where I_0 is the incident beam intensity. Eq.(2.1) will then simplify to following form:

$$I_i = I_0 exp\{-\int_{L_i} \mu(x_1,x_2,x_3,)dl\} \tag{2.2}$$

De Man in (De Man *et al.*, 2001) proposed a method which accounts for the polychromatic nature of the source and the results was shown to be highly effective in artifact suppression. In

this work, the energy dependence of the attenuation coefficient is modeled by decomposing it into photoelectric component and Compton scatter component.

Moreover, the measurements from detectors are treated independently for both components. Therefore, one can use the same projection operator developed for the monochromatic case for this model. Consequently, in developing algorithms one can use the monochromatic model to benefit from the simplicity of this model and the results can then be extended to polychromatic model using the work of (De Man *et al*., 2001).

In the next section, the linear forward model of X-ray tomography will be provided.

2.2.2 Linear Forward Model

The goal of tomographic reconstruction is to estimate the linear attenuation coefficients $\mu(\vec{x})$ from a set of realizations ($\{y_i = Y_i\}_{i=1}^{N_Y}$, where N_Y is the product of number of detectors and number of projection angles) which are obtained through data acquisition process.

It is known that the realizations or the measurement data are discrete, moreover, the reconstructed image will be represented on a digital display and will have finite number of pixels. Consequently, it is natural to represent $\boldsymbol{\mu}(\vec{x})$ in a discrete form. The discretized model of eq. (2.2) can then be written as following:

$$I_i = \int I_0(\mathscr{E}) exp\{-\mathbf{p}_i^T \boldsymbol{\mu}(x_1, x_2, x_3, \mathscr{E}) dl\} d\mathscr{E} \tag{2.3}$$

where $\mathbf{p}_i$ is a vector and represents the contribution of i-th ray through the pixels which this ray passes through. Consequently, the projection data measured by detectors can be written as:

$$\underline{\mathbf{I}} = \int I_0(\mathscr{E}) exp\{-\mathbf{P}\boldsymbol{\mu}(x_1, x_2, x_3, \mathscr{E}) dl\} d\mathscr{E} \tag{2.4}$$

$\mathbf{P}$ is the discrete projection operator and is made of $\mathbf{p}_i$ vectors. This operator approximates the integrals over the discrete object and in tomography problems is very large and sparse.

Now assuming that the X-ray attenuation coefficients are independent of the X-ray source energy, using logarithmic transformation one can compensates for the nonlinearity of Beer's law and simplify the problem as following:

$$\mathbf{y} \triangleq log \frac{I_0}{\underline{\mathbf{I}}} = \mathbf{P}\boldsymbol{\mu} \tag{2.5}$$

where $\mathbf{y}$ represents the sinogram.

Assuming that the noise can be modeled as additive noise in tomography problems, the linear deterministic imaging model can then be expressed by following Gaussian model:

$$\mathbf{y} = \mathbf{P}\boldsymbol{\mu} + \mathbf{b} \tag{2.6}$$

where $\mathbf{b} \rightarrow N(0,\mathbf{\Sigma}_b)$ represents noise and $\mathbf{\Sigma}_b$ is the variance of noise. The characteristics of $\mathbf{\Sigma}_b$ will be discussed in section 2.2.2.2 where the noise model is studied.

In order to estimate the linear attenuation coefficients of the object using above relationship, one needs to first compute the projection matrix $\mathbf{P}$.

2.2.2.1 Projection Matrix Calculation

It should be noted that, in practice detectors which are used in CT imaging are not perfect. By perfect we mean that they are not infinitely small and therefore the number of rays reaching a detector is more than one; the infinitesimal line integral in (2.1) and (2.2) is an approximation. To overcome this problem, one can use multiple rays for modeling the detectors, while calculating the coefficients of $\mathbf{P}$. The number of rays U, reaching each detector will then be chosen based on the type of scanner used for data acquisition. The average of U rays will then be used for calculating each coefficient of projection operator.

Different methods have been introduced for calculating the projection operator coefficients. This includes: pixel-driven, ray-driven and distance-driven. For detailed explanation on how each method works and their advantages and disadvantages see Appendix A. Ray-driven is usually preferred among these methods due to its simplicity and compatibility for hardware implementation.

2.2.2.2 Noise

Although one may be able to detect the high-contrast objects in an image with high noise-level, this will not be easy for objects with low-contrast.

The most probable source for the noise added to the projection data in CT imaging is the Poisson noise due to the quantum nature of the X-ray photons (Guan et Gordon, 1996). Poisson noise is due to the low photon counts in X-ray CT. This means that as photon counts approaches zero, the noise increases rapidly and as a result the maximum attenuation has a higher effect than the average attenuation in Poisson noise.

In medical imaging, the dose in radiography is kept as low as possible because of health issues. This causes the X-ray quantum noise to become more apparent in the sinogram. This is because the quantum noise is dependent on the average number of X-rays reaching detectors, therefore decreasing the dose causes a higher random variation in the number of X-rays reaching detectors and consequently affects the quality of image. This is known to be a fundamental limit in the radiography and requires including the appropriate noise model in the reconstruction of image (Mudry *et al.*, 2003).

By Central Limit Theorem, Poisson distribution of the photon counts can be approximated by Gaussian distribution for large average photon counts. Use of Gaussian distribution is preferred due

to the simplicity of the model and allowing quadratic objective function to be easily solved using linear optimization methods. Therefore additive noise in Gaussian model (2.6) can be modeled as either Gaussian with independent variance (referred to as Gaussian noise here) or with variance dependent on the photon counts to account for the Poisson distribution of quantum noise (referred to as Poisson noise here). Using Gaussian model for noise has a simpler form and variance of noise can be represented as $\mathbf{\Sigma}_b = \sigma\mathbf{I}$, where σ is the standard deviation of noise. For Poisson noise, one can use the Gaussian approximation developed in (Sauer et Bouman, 1993) as following:

$$\mathbf{\Sigma}_b = diag\{exp(-y_1), exp(-y_2), \ldots, exp(-y_{N_Y})\} \tag{2.7}$$

$\mathbf{\Sigma}_b$ in the above equation represents the variance of noise in (2.6) and is dependent on the photon counts ($\underline{\mathbf{I}}$) which has been used for obtaining the sinogram $\mathbf{y}$ in 2.5.

2.3 Reconstruction problem

There are some physical problems associated with imaging using CT scanners. One of the main sources of inaccuracy comes from the fact that the number of photons counted is limited. This limitation could be due to the source's strength, motion of the patient during photon transmission, absorption of some photons during the transmission of X-ray and limited dose of X-ray for avoiding harm to patient and minimizing the scan duration (Fessler, 2000). Moreover, the projection data is usually noisy and for a high quality reconstruction one needs to consider the noise in the model.

In this section different approaches which are used in practice for tomography image reconstruction will be discussed briefly. These methods can be categorized into two groups: analytical and statistical approaches. Limitations and advantages of each method will be reviewed in following subsections.

2.3.1 Analytical approach to reconstruction

The conventional method used in reconstructing the images in X-ray CT is FBP which uses the Fourier slice theorem for reconstructing the image. This method benefits from the fast calculation property of the fast Fourier transform (FFT) technique. However, the quality of images obtained using this technique is unsatisfactory in some practical cases, e.g., noisy data measurements (Elbakri et Fessler, 2002), low dosage medical imaging (Mc Kinnon et Bates, 1981) and in nondestructive testing of material with widely varying densities (Sanderson, 1979). This is because of simplifications made in the model used for reconstruction and not considering the measurement noise in the model. In this method a filter is applied to the sinogram and the result is then backprojected.

Assuming the $\mathbf{F}_{FBP}$ represent the filter, this can be written as following:

$$\hat{\boldsymbol{\mu}} = \mathbf{P}^T \mathbf{F}_{FBP} \mathbf{y} \tag{2.8}$$

This model can be seen as a simplified version of the linear deterministic model (2.6). As can be seen in (2.8), this approach does not consider the noise in the model and consequently quality of reconstructed images is very sensitive to reduction of radiation dose. This method is also known to be poorly suited to nonstandard imaging geometries, i.e., truncated fan-beam or cone-beam (Fessler, 2000).

2.3.2 Statistical approach to reconstruction

In tomography problems and specifically medical imaging where low-dosage of X-rays is necessary, noise is of primary concern. Therefore, the image reconstruction problem is usually treated as a statistical estimation problem. Statistical methods can be used to overcome some of the problems discussed in Section 2.3.1. This is mainly because in this technique, the model derived in (2.6) is used and so the noise is considered in the model. This will reduce the noise level in the reconstructed images significantly. Moreover, this method can improve the image quality by incorporating some prior knowledge on images and has the capability to model the energy-dependency of the beams and the potential to be used for different scanning geometries. As a result, it has proved to outperform FBP in CT (De Man *et al.*, 2000; Wang *et al.*, 1999) by noise reduction, resolution improvement, and in some cases, artifact suppression. The main drawbacks of these methods is known to be their long reconstruction time and the high memory need.

In the next section, reformulation of statistical methods in an estimation framework will be discussed briefly. By choosing this framework properly, one can then obtain a reconstruction algorithm which is more robust to noise and artifacts.

2.3.2.1 Estimation

A statistical problem can be solved using estimation techniques. Sauer and Bouman in (Sauer et Bouman, 1993) and Fessler in (Fessler, 1994) showed that one can use a quadratic approximation to the log-likelihood function for transmission and emission tomography problems respectively to simplify the calculations. The objective function to be minimized, then consists of a weighted least-square term and corresponds to that of a *Maximum likelihood* (ML) estimator.

A regularization function can also be used to add prior information on the distribution of the image (Fessler, 2000). Adding the prior distribution of image to regularize the optimization problem ensures that a unique solution always exists and the problem can be solved using algorithms

designed for inverse problems (Tikhonov et Arsenin, 1977). This will also reduce the variances, which are mainly due to the noise and makes the system robust to small changes and noise.

The objective function with weighted least-square and the regularization function, subject to non-negativity constraint, will then have following form:

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\geq 0}{\operatorname{argmin}_{\boldsymbol{\mu}}} \ \Phi(\boldsymbol{\mu}) \quad , \quad \Phi(\boldsymbol{\mu}) \triangleq \frac{1}{2}\|\mathbf{y}-\mathbf{P}\boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}_b} + \lambda R(\boldsymbol{\mu}) \tag{2.9}$$

where λ is the scalar, which controls the trade-off between fidelity of solution to measured data and prior information and is called regularization parameter. $R(\boldsymbol{\mu})$ is the regularization function, the choice of regularization function will be discussed in detail in Section 2.3.2.2.

The noise model is included in the above model with Gaussian distribution and variance $\boldsymbol{\Sigma}_b$. This weighting matrix can be chosen as explained in section 2.2.2.2, to be either constant or dependent on the photon counts.

The above objective function with regularization function lies in the *maximum a posteriori* (MAP) estimation from Bayesian family. This method can be seen as a regularized ML with better performance compared to ML.

2.3.2.2 Regularization Function

It is considered that a desirable attenuation map is one which is a piece-wise smooth function. This means that an appropriate penalty function $R(\mu)$, is the one which discourages images that are too 'rough' (Fessler, 2000). Assuming that the object is discretized in Cartesian coordinates, the simplest penalty function with such properties which penalizes the neighboring pixels and considers the discrepancies between neighboring pixel values can be shows as below:

$$R(\boldsymbol{\mu}) = \frac{1}{2}\sum_{j=1}^{N_p}\sum_{k=1}^{N_p}\omega_{jk}\psi(\mu_j-\mu_k) \tag{2.10}$$

where $\omega_{jk} = \omega_{kj}$. The rationale for this, is to penalize some numerical approximation to the gradient of μ. Therefore, if one wants to consider the first differences only, $\omega_{jk} = 1$ for four horizontal and vertical pixels, $\omega_{jk} = 1/\sqrt{2}$ for diagonal neighboring pixels, and $\omega_{jk} = 0$ otherwise. ψ is a function which assigns some cost to neighboring pixels $\mu_j - \mu_k$ and needs to be symmetric and differentiable.

Most of the regularization functions used in tomographic reconstruction can be represented in following general form (Fessler, 2000):

$$R(\boldsymbol{\mu}) = \frac{1}{2}\sum_{k=1}^{K}\psi_k([\mathbf{D}\mu]_k) \tag{2.11}$$

where $\mathbf{D}$ is the matrix of penalty and is $K \times N_p$ and

$$[\mathbf{D}\mu]_k = \sum_{j=1}^{N_p} d_{kj}\mu_j \tag{2.12}$$

Note that equation 2.10 can be seen as a special case of equation 2.11 with $K \approx 2N_p$.

In general, regularization functions can be categorized into three groups: quadratic-convex, non-quadratic-convex and non-quadratic-nonconvex. The standard minimization techniques are guaranteed to convergence to the global solution (minima), if the objective function is both differentiable and convex (or locally convex). Quadratic functions lead to simple calculations, however, they are not the best option for preserving the sharp edges in neighboring pixels. For this purpose, non-quadratic function can be used to benefit from their properties.

Different functions have been used in the literature, to name a few: Gaussian Markov (quadratic-convex), L2-L0 (non-quadratic-nonconvex) and log prior distribution (from Kullback pseudo-distance)[1]. They either suffer from not allowing sharp edges or not being globally convergent.

Generalized Gaussian Markov random Field (GGMRF), proposed in (Bouman et Sauer, 1993a), and L2-L1 are two examples of non-quadratic regularization functions with convex property. These functions are found efficient in tomographic reconstruction and povide interesting results. However, using these functions will increase the complexity of the problem to be solved compared to when quadratic regularization is used.

L2-L1, a variant of the Huber function, consists of a branch of hyperbola and allows quadratic and linear behavior. It is defined as: $\psi(t) = \sqrt{t^2 + \delta^2} - \delta$, where δ adjusts the transition between the quadratic and linear behavior of ψ. Among the available functions used for regularization, this function is found efficient in providing a good trade-off between noise and resolution (Menvielle, 2004),(Gendron *et al.*, 2008), (Hamelin *et al.*, 2008) ,(Hamelin *et al.*, 2010a) with a reasonable degree of complexity.

2.4 Optimization algorithms

For reconstructing images using statistical methods, neglecting the non-negative constraint, one needs to find the value of $\hat{\mu}$ for which the gradient of cost function Φ, becomes zero, numerically. The gradient of objective function is represented as following:

$$\nabla\Phi(\boldsymbol{\mu}) = -\mathbf{P}^T\boldsymbol{\Sigma}(\mathbf{y} - \mathbf{P}\boldsymbol{\mu}) + \lambda\nabla R(\boldsymbol{\mu}) \tag{2.13}$$

[1] This function is non-quadratic and is convex if the composition of logarithmic function and the function of interest (prior distribution) is convex

In the above relationship, the multiplication of projection by any vector is called projection ($\mathbf{P}\boldsymbol{\mu}$) and the multiplication of its transpose with any vector is called backprojection operation. These operations need to be calculated repeatedly when the problem is solved iteratively.

For non-quadratic penalty functions $R(\mu)$, it is not possible to obtain an explicit solution to (2.13). Even, if one uses the quadratic function for the penalty term , the closed-form solution will have the following form:

$$\hat{\boldsymbol{\mu}} = (\mathbf{P}^T\boldsymbol{\Sigma}\mathbf{P} + \lambda\mathbf{D}^T\mathbf{D})^{-1}\mathbf{P}^T\boldsymbol{\Sigma}\mathbf{y} \tag{2.14}$$

and requires calculating the inverse of normal matrix. In tomography problems, considering the large size of data, this is not possible and so calculating this problem directly is impractical. For such cases, iterative methods can be used to find the minimizer $\hat{\mu}$ of this objective function, iteratively.

Using numerical techniques, an image is reconstructed by minimizing Φ iteratively. The complexity of minimization of the cost function is dependent on several factors. These include: the data formation and the regularization function used.

Different factors needs to be taken into account when designing/choosing an algorithm. Some of these which are of special concern in tomography imaging are listed below:

- Non-negativity constraint
- Convergence rate
- Computation time per iteration and the possibility for parallelization
- Memory requirement

In this work, we are interested in minimizing the objective functions $\Phi(\boldsymbol{\mu})$, which consists of a penalized least square term and a convex, twice differentiable regularization function $R(\boldsymbol{\mu})$.

In the literature, different numerical methods have been suggested and developed for X-ray CT reconstruction. Efficiency of each method varies depending on the application, required quality, available memory footprint and acceptable processing time. Below, a brief review of the present works is given and advantages and disadvantages of each are discussed. Some of these algorithms are specifically designed for tomography problem by considering the special structure of the objective function. Performance of these methods are compared with that of general purpose algorithms.

2.4.1 A review on the numerical method used for tomography reconstruction

Different approaches have been followed for minimizing the cost function for emission and transmission tomography. In general, these works can be studied in two different way: one is based on how they update the pixel values and the other based on the family of iterative methods used. In

this work the first approach is used. This is because this approach has been followed by researchers for developing algorithms compatible with the needs of the tomography problem. In the sequel, the affiliation of these works with different families of iterative methods will be also discussed.

The numerical methods used for solving tomography problems can be divided into two groups based on how they update the pixel values: 1) Those which update all pixels simultaneously and, 2) Those which update individual pixels iteratively (Qi et Leahy, 2006).

Gradient-based technique from family of iterative methods, is one of the common techniques used for image reconstruction which update all pixels simultaneously. Steepest ascent and conjugate gradient (CG) (Kaufman, 1987) with linear convergence, are two widely used methods of this family.

The problem with these methods is that they do not account for non-negative constraints.To account for non-negativity constraints, coordinate-wise methods have been introduced which update individual pixels sequentially. This method can be used for both quadratic and non-quadratic cost functions, with more complicated calculations involved for the latter. Comparing coordinate-wise method with methods which update pixels simultaneously, there are slightly more per iteration computations involved in this approach. The other disadvantage is that they hinder the possibility of parallelization. A known example of this group is iterative coordinate descent (ICD) used in (Sauer et Bouman, 1993) and (Bouman et Sauer, 1996) for transmission tomography and in (Fessler, 1994) for emission tomography.

Ref. (Sauer et Bouman, 1993) used the ICD method with a quadratic regularization function and compared its performance with two other methods of gradient ascent (GA) and CG from gradient based family. It was illustrated that CG overtook ICD for the regularized case in terms of convergence speed. Bouman and Souer then introduced the ICD/Newton-Raphson (Bouman et Sauer, 1996) to increase the convergence speed, however this method was criticized for not guaranteeing the global convergence.

Fessler et al. (Fessler *et al*., 1997) tried to solve this problem by introducing grouped-coordinate method and using function substitution. This method has a faster convergence speed, allows parallelization and also benefits from considering the non-negativity constraint, but is hard to implement. The same problem was encountered in (Zheng *et al*., 2000).

Separable paraboloidal surrogates (SPS) was then used in (Erdoğan et Fessler, 1999a) for minimizing the cost function and lead to an increase in the convergence speed by substituting the cost function by a function easier to minimize. However, the convergence of this algorithm is fast in the first few iterations and then becomes slow. An attempt to further improve the convergence rate was then followed by using ordered subsets with SPS family (OS-SPS) (Erdoğan et Fessler, 1999b). However, the non convergent property of OS-SPS hinders its use in clinical image reconstruction where reliable results are required. Some works have been done in this area to make the method

convergent, i.e., transmission incremental optimization transfer (TRIOT) (Ahn *et al*., 2006) to name one. However, this resulted in a reduced speed of convergence compared to OS-SPS. Another problem with TRIOT is that it requires more memory space and needs to save all previous iteration to calculate the next iteration.

Performance of the four algorithms of ICD, CG, OS and TRIOT, which were found more efficient in image reconstruction were compared in (De Man *et al*., 2005). It was illustrated that all of the methods gave similar quality images when converged. This is because the increase in computation time per iteration was compensated by a similar decrease in the number of iterations.

The numerical methods used in above discussed works are all from the gradient-based family. CG can easily adopt to nonlinear optimization, however, as discussed above, they suffer from low convergence speed. Moreover, they do not account for non-negativity constraint and require an additional step to address this problem. This will further decrease their convergence speed. On the other hand, iterative methods from Quasi-Newton family can be used to address these problems and obtain a super-linear convergence. L-BFGS-B, an example of Quasi-Newton family (Zhu *et al*., 1997) uses an approximation to the Hessian matrix for every iteration instead of calculating the exact Hessian. It also has the advantage of considering the non-negativity constraint compared to CG and other gradient-based methods without constraints. This algorithm was used by Hamelin in (Hamelin *et al*., 2008) and its efficiency in improving convergence was verified for tomography reconstruction.

Ref. (Hamelin *et al*., 2010a) provides a good comparison between four efficient numerical methods used for tomography reconstruction. This includes two methods from SPS family (OS-SPS and TRIOT) which are specially developed for CT reconstruction and two general solver (L-BFGS and IPOPT). L-BFGS-B and IPOPT, from Quasi-Newton family with limited memory and Hessian-free property, make their use in large clinical problems possible. Comparing the convergence speed of the four methods, OS-SPS had the highest speed followed by two methods from general solver family. However, it did not reach the stopping criteria as expected. Simulation and experimental results in (Hamelin *et al*., 2010a), proved that for noisy data with high weight of penalty function, TRIOT gives the fastest result whereas for low noise level in data sets, general solvers work better.

This brings us to conclude that among the methods discussed in this section, the SPS method from gradient-based family and, the CG and L-BFGS-B from the general purpose algorithms outperform the other iterative methods for image reconstruction. The SPS method is efficient when using the Poisson log-likelihood formulations (naturally used in PET and SPECT) whereas general purpose algorithms are more general and have a simple structure to be used. Comparing the CG and the L-BFGS-B, the CG method (the nonlinear version) can be seen as a special case of L-BFGS with very limited memory (See chapter 6 (Nocedal et Wright, 1999)). Therefore, a better convergence rate is expected for L-BFGS-B. Moreover, L-BFGS-B has the bound-constraint property which is

implemented efficiently and using it does not reduce the convergence speed considerably (as oppose to other methods with bound constraint). In Appendix B a brief review on implementation of these methods (gradient based and Quasi-Newton family) and of their properties is provided.

To summarize, for tomography image reconstruction, where the size of data is large and image needs to be provided in a short period of time, convergence is of great importance and the numerical algorithm needs to be chosen appropriately. Consequently, the algorithm to be used needs to have the following properties: allow parallel implementation, provide reliable result to be able to be used for clinical purposes, and have a moderate storage need. Using the above discussion on the comparison of performance of iterative methods used in literature for solving tomography problems, we can conclude that L-BFGS-B from the quasi-Newton family provides a good trade-off between the memory need and convergence speed. This method also accounts for the required constraints and can be used for solving nonlinear optimization problems.

2.4.2 Conditioning and preconditioners

As discussed, in solving tomography problems, given the large size of the estimation problem it is necessary to use iterative methods. The convergence speed of the iterative methods depends on two essential factors: condition number and eigenvalue spectrum of the normal matrix (the normal matrix will be defined later). In this section, these points will be studied.

Generally, in solving optimization problems, numerical methods are used to find the global minimum of an objective function (Φ). In our problem of interest, with the objective function given in (2.9), this is done by finding $\hat{\mu}$ for which the gradient $\nabla\Phi(\boldsymbol{\mu})$ represented in (2.13) becomes zero. If $\nabla\Phi(\boldsymbol{\mu}) = 0$ is linear and has a closed form solution, then the optimization problem can be represented as a large-scale linear system with n-equations as following:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.15}$$

with $\mathbf{A}$ being the normal matrix. Iterative methods work by decreasing the residual ($\mathbf{r}^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}$) at every iteration by finding a sequence of solutions ($\mathbf{x}^k$).

The condition number of a matrix is a measure of how well-conditioned a matrix is, and for a non-singular matrix $\mathbf{A}$, it is defined as follows (Chen, 2005):

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{2.16}$$

with $\|.\|$ being the matrix norm[1]. For a symmetric, nonsingular matrix, this relationship reduces to:

$$\kappa(\mathbf{A}) = \frac{\mid \lambda \mid_{max}}{\mid \lambda \mid_{min}}$$

where $\mid \lambda \mid_{max}$ and $\mid \lambda \mid_{min}$ correspond to largest and smallest eigenvalue of system matrix respectively, and $\kappa(\mathbf{A}) \geq 1$. The condition number is one of the factors which strongly affects the convergence speed of numerical methods. Ideally, κ is equal to 1 and problem converges in one iteration only. Another factor which is essential in the speed of convergence for solving a system of equations using iterative methods, is how wide the eigenvalues are spread. For linear CG method, it is known that the optimization problem converges in maximum n number of iterations where n represents the order of system or the number of eignevalues. However, for the problem with clustered eigenvalues the maximum number of iterations is equal to the number of clusters. The same reasoning can be used for other numerical methods (Nocedal et Wright, 1999). Therefore, we can conclude that solving a system with widely spread eigenvalues can be much slower than the one with same condition number but less spread eigenvalues.

Generally, the condition number for tomography problems is far from unity or the eigenvalues are spread in a wide range, therefore, iterative methods require several iterations to converge and so the reconstruction time is long for statistical methods. The condition number and the eigenvalue spectrum of a system of equations can be improved by designing efficient preconditioners. Preconditioning and some practical forms of this technique will be discussed in the next section.

2.4.2.1 General strategy for preconditioning

Consider the linear system defined in (2.15) with $\mathbf{A}$ being an invertible square matrix. Given an invertible matrix $\mathbf{\Pi}$ of same order, the following system of equations gives the same solution as (2.15):

$$\mathbf{\Pi}^{-1}\mathbf{A}\mathbf{x} = \mathbf{\Pi}^{-1}\mathbf{b} \tag{2.17}$$

$\mathbf{\Pi}^{-1}$ is called the preconditioning matrix and equation (2.17) is known to be a preconditioned system. Here the preconditioning matrix is multiplied by the system matrix from the left hand side. Therefore, this preconditioner is called left-preconditioner. In the preconditioned system (2.17), $\mathbf{\Pi}$ needs to be chosen such that solving system of equations in (2.17) will be easier than that of

[1] A matrix norm is a vector norm on $\mathbb{R}^{m\times n}$. That is, if $\|\mathbf{A}\|$ denotes the norm of the matrix $\mathbf{A}$, then (Chen, 2005):

- $\|\mathbf{A}\| \geq 0$ iff $\mathbf{A} = 0$,
- $\|\alpha\mathbf{A}\| = \mid \alpha \mid \|\mathbf{A}\|$ for all α in $\mathbb{R}$ and all matrices $\mathbf{A}$ in $\mathbb{R}^{m\times n}$,
- $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ for all matrices $\mathbf{A}$ and $\mathbf{B}$ in $\mathbb{R}^{m\times n}$.

(2.15). An ideal preconditioner is the one which is the exact inverse of system matrix, $\mathbf{A}$ in (2.15). However, calculation of the exact inverse of system matrix usually has the same cost as solving the problem without preconditioner.

Preconditioners can improve the convergence speed of iterative methods by decreasing the condition number of the problem. The convergence of an iterative method will be very dependent on the spectral properties of the new matrix $\mathbf{\Pi}^{-1}\mathbf{A}$ and with choosing $\mathbf{\Pi}$ appropriately, will be better than that of $\mathbf{A}$. In practice, we do not calculate the $\mathbf{\Pi}^{-1}$ and we only need to know the matrix-vector product. For a preconditioned system of equation, the matrix-vector operation corresponds to:

$$\mathbf{w} = \mathbf{\Pi}^{-1}\mathbf{A}\mathbf{x} \quad \Longleftrightarrow \quad \mathbf{\Pi}\mathbf{w} = \mathbf{A}\mathbf{x} \tag{2.18}$$

which $\mathbf{\Pi}\mathbf{w} = \mathbf{c}$ can be easily solved[1].

Similarly, $\mathbf{\Pi}$ is a right-preconditioner matrix if (2.15) is solved as following:

$$\mathbf{A}\mathbf{\Pi}^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{\Pi}^{-1}\mathbf{y} \tag{2.19}$$

In (2.19), preconditioner is applied using a change of variable $\mathbf{x} = \mathbf{\Pi}^{-1}\mathbf{y}$.

For choosing a preconditioner for a system which will be solved iteratively, one needs to assure that calculation of this matrix is not expensive and the extra matrix-vector product calculations required for this technique will not destroy the gain obtained by transforming the problem to a simpler one. This means that in designing practical preconditioners for solving large size problems, there are always some approximations involved. Consequently, design of efficient preconditioners will be a trade-off between the extra calculation per iteration, the memory need and the increase in the convergence speed.

In general, there are two approaches in devising a preconditioner: one is to find a matrix $\mathbf{\Pi}$ which approximates the system matrix $\mathbf{A}$, and then solving the problem with the inverse of this matrix multiplied from left, right or both sides, by (2.15) is easier than the one with $\mathbf{A}$. This is called the **forward type**. Examples of this category are diagonal preconditioners which preserve the elements in diagonal of system matrix and the block-approximations to the normal matrix. The other approach is to find a matrix $\mathbf{\Pi}$ which approximates the $\mathbf{A}^{-1}$ and multiplying this matrix by system of equations (2.15), will make solving the problem easier. This is called the **inverse type**.

Therefore, preconditioners can be categorized as below (Chen, 2005):

1. **Forward type**: $\mathbf{\Pi} \approx \mathbf{A}$

   Left $\mathbf{\Pi}^{-1}\mathbf{A}\mathbf{x} = \mathbf{\Pi}^{-1}\mathbf{b}$

[1] Different methods are available for solving a system of equations. This includes LU decomposition, Newton-Schulz-Hotelling, Gauss-Jordan decomposition and QR decomposition, to name a few (Chen, 2005). In this work Cholesky decomposition will be used. The justifications for this choice will be provided in the relevant sections.

Right $\mathbf{A\Pi^{-1}y = b},\ \mathbf{x = \Pi^{-1}y}$

Mixed $\mathbf{\Pi_2^{-1}A\Pi_1^{-1}y = \Pi_2^{-1}b},\ \mathbf{x = \Pi_1^{-1}y}$

2. **Inverse type**: $\mathbf{\Pi \approx A^{-1}}$

   Left $\mathbf{\Pi Ax = \Pi b}$

   Right $\mathbf{A\Pi y = b},\ \mathbf{x = \Pi y}$

   Mixed $\mathbf{\Pi_2 A\Pi_1 y = \Pi_2 b},\ \mathbf{x = \Pi_1 y}$

with $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ being right and left preconditioners and $\mathbf{\Pi_2\Pi_1 = \Pi}$.

Remark 2.4.2.1: Assuming that the regularization function used in (2.9) is quadratic, the preconditioner can be easily applied to many numerical methods using a simple change of variable. The objective function to be solved with preconditioner ($\mathbf{\Pi}$) and using $\boldsymbol{\mu} = \mathbf{\Pi u}$, will then be:

$$\Phi(\mathbf{\Pi u}) = \frac{1}{2}(\mathbf{y - P\Pi u})^T(\mathbf{y - P\Pi u}) + \lambda(\mathbf{\Pi D u})^T(\mathbf{\Pi D u}) \tag{2.20}$$

and its gradient will have following form:

$$\nabla\Phi(\mathbf{\Pi u}) = -\mathbf{\Pi}^T\mathbf{P}^T(\mathbf{y - P\Pi u}) + \lambda\mathbf{\Pi}^T\mathbf{D}^T\mathbf{D\Pi u} \tag{2.21}$$

The hessian of this function or its normal matrix will then be:

$$\begin{aligned}\nabla^2\Phi(\mathbf{\Pi u}) &= \mathbf{\Pi}^T\mathbf{P}^T\mathbf{P\Pi}^T + \lambda\mathbf{\Pi}^T\mathbf{D}^T\mathbf{D\Pi} \\ &= \mathbf{\Pi}^T\{\mathbf{P}^T\mathbf{P} + \lambda\mathbf{D}^T\mathbf{D}\}\mathbf{\Pi}\end{aligned} \tag{2.22}$$

Equation (2.22) suggests that preconditioning by change of variable for any numerical method, results in a mixed preconditioner, and we need to find a factorization of an approximation to $\mathbf{A}$ in the form $\mathbf{\Pi}^T\mathbf{\Pi}$.

2.4.2.2 Some practical preconditioners

As discussed in Section 2.4.1, most of the iterative methods used in the literature for image reconstruction are gradient-based techniques. This technique is found efficient and can be used to solve systems with quadratic or nonquadratic convex cost function. Moreover it benefits from simplicity and possibility for parallel implementation (Fessler et Booth, 1999). However, the main disadvantage of this method is the low convergence speed.

Different types of preconditioners have been designed and attempted to be used for tomography problems with performance varying from problem to problem (Fessler et Booth, 1999; Gendron

et al., 2008). Diagonal preconditioners, are known as the most simple and still efficient one, however, they are not the best for imaging problems. Moreover, the condition number of tomography systems with these preconditioners can be far from unity and therefore convergence speed may remain slow.

Circulant preconditioners, provide a better result for imaging problems however, they require Hessian matrix to be shift-invariant and block diagonal. This is not always true due to the non-uniform noise variance of photons and limits the efficiency of this type of preconditioners.

Important general purpose preconditioners are: incomplete Cholesky and banded preconditioners (Nocedal et Wright, 1999), to name a few. Incomplete Cholesky decomposition is found very efficient for preconditioning when the system of equations is large, Hermitian, definite nonnegative and has a sparse structure. In this method, the same principle as the exact Cholesky is used.

In exact Cholesky, a lower triangular matrix $\mathbf{L}$ is found such that it satisfies: $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. The lower triangular matrix in the exact Cholesky is usually full and therefore is not efficient for preconditioning the large sparse reconstruction problems. In contrast, incomplete Cholesky uses a sparse triangular matrix $\tilde{\mathbf{L}}$ with same sparsity level as $\mathbf{A}$ which satisfies $\mathbf{A} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$. In this technique any entry in $\tilde{\mathbf{L}}$ is set to zero if the corresponding entry in $\mathbf{A}$ is zero. This reduces the memory requirement considerably. The preconditioning matrix will then be: $\mathbf{\Pi} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$.

Using sparse approximations for any positive-definite matrix may destroy the positive-definite property for some thresholds. However, the incomplete Cholesky factorization always exists for diagonally dominant matrices. Therefore, for such cases one can add a small fraction (α) of diagonal values to this matrix to make it diagonally dominant. By doing this we will make another approximation but this will also increase the stability. The optimal value of α needs to be calculated experimentally and by try and error.

The sparse triangular structure of incomplete Cholesky technique allows fast calculation of matrix-vector product required for preconditioning and is desired when used with iterative methods. This will be discussed in more detail in chapter 5 where appropriate preconditioners are designed.

2.4.2.3 A review on the preconditioning techniques used in CT reconstruction

In general, when the object is represented in the standard Cartesian grid, the projection operator has a limited degree of symmetry. This limits the redundancy in the normal matrix and makes designing proper preconditioners hard. Consequently, not many works for attempting to design preconditioners have been reported in the literature. Below two of the limited works done for using preconditioner for tomography problems are discussed.

In (Fessler et Booth, 1999), Fessler suggested a new preconditioner which is a combination of diagonal/circular preconditioners and claimed that the new preconditioner is more efficient in terms of convergence rate for general shift-variant cases. This claim was proved by applying it to CG and

comparing the results with those from the ICD method in (Bouman et Sauer, 1993b). The resulting improvement was obtained at the cost of an increase in the computation time. This preconditioner was criticized later for its implementation complexity in (Ramani et Fessler, 2011).

Gendron in (Gendron *et al.*, 2008) also tried to find an efficient preconditioner by designing circulant at zero (CZ) and circulant at the current point (CC) preconditioners. The performance of these preconditioners was compared with that of (Fessler et Booth, 1999). Simulations showed that for the high level of noise, although some improvements is achieved in computation time, the results are not satisfactory at the presence of small metal objects.

From the above discussion, we can see that designing efficient and practically usable preconditioners is not easy. Preconditioners are designed by making a heuristic approximation to the normal matrix. Consequently, if a data formation can be used for representing the object which result in having a specially structured normal matrix, a more efficient preconditioner with less approximation can be designed.

In the next section, different types of object representation used in the literature will be discussed and the advantages and disadvantages of each method will be underlined.

2.5 Object Representation

In general, the most common form used for discretizing the object to be reconstructed is to use the regularly spaced Cartesian grid. Using this model, the coefficients of projection operator vary from one projection angle to another. Even if symmetry properties and rotational invariance are taken into account, storage of the whole projection operator is still impossible for clinical 3D CT data. The general solution to this problem is to calculate a significant part of the entries of the projection matrix on-the-fly. Although, this approach will address the memory limit, it will result in a loss of computational efficiency as it increases the computation time. Alternative approaches have been proposed to overcome these difficulties. These approaches will be briefly discussed below and their efficiency will be underlined.

2.5.1 Targeted CT

In this approach, it is assumed that high quality reconstruction is required only in a small region of the image. This small region is called region of interest (ROI) and it is attempted to obtain a high quality reconstruction for this region. To do this object is splitted into two regions: the high resolution ROI and the coarse background. Both regions are then discretized on Cartesian grids with different stepsizes(Hamelin *et al.*, 2010b). Although, interesting results are observed using this approach, its practical use is hindered by the heavy computation time involved and the difficulty for implementation.

In (Brankov *et al*., 2004), another approach is suggested for discretizing the image to obtain a high quality reconstruction in a small region of the image. In this work, the object is discretized on an adaptive irregular mesh with a high vertex density in the ROI and low vertex density in the background. This method is found difficult to implement moreover, there are some questions about the mesh construction practical computation of the projection operator. The variable pixel sizes in this type of discretization will also affect the conditioning of the problem and so decrease the convergence speed.

2.5.2 Rotation-based approaches

The general idea in the rotation-based approaches is to reduce the memory need for storing the projection matrix by storing a partial projection operator only. The partial projection operator will correspond to a single or a small number of projection angles and the object will be rotated appropriately at each projection angle to perform the projection and backprojection operations. It should be noted that in this approach, the object must be represented on a geometrically fixed grid while it is being rotated. Using this technique will result in a dramatic reduction in memory need for storing the projection operator as only a partial part of this matrix needs to be stored. This is specifically true for X-ray CT where the number of projection angles is large.

One possibility in this approach, is to discretize the object on a regular Cartesian grid (Zeng et Gullberg, 1992; Zeng *et al*., 1994). The image volume will then be rotated at each projection angle so that the front face of the voxelized cube is kept parallel to the detection plane. Rotation of the object in this method at every projection angle will require interpolation and so introduces some approximation. Moreover, performing interpolation at every projection angle is time consuming and having to store the coefficients of rotation matrices will reduce the gain achieved by using this approach.

The main bottleneck of this approach however is known to be the approximations generated by rotating the object. These approximations may make the projection and backprojection operators not exact transpose of each other and consequently affect the convergence. These explain why this method have not been widely used (Goussard *et al*., 2013) .

2.5.3 Sector-invariant approaches

Another approach for discetizing the object to be reconstructed is to use the sector-invariant manner. In this approach natural symmetry of the scanner's geometry is taken advantage of, by discretizing image into some sectors each corresponding to a projection angle of the partial projection operator. Complete projection can then be computed by repeated application of the partial projection operator to the sector-wise rotation of the object for each angle. Note that sector-wise rotation of the object

is a simple circular shift of the vector holding the discretized object samples and so rotation can be performed easily for each projection angle. Using this approach allows pre-calculating and storage of the projection operator easily.

Different sector tessellation schemes have been proposed and used for image reconstruction mainly in PET reconstruction (Mora et Rafecas, 2006; Jian *et al*., 2007; Mora *et al*., 2008; Rodriguez-Alvarez *et al*., 2011). The complexity of the proposed tessellation schemes however make the computation of the partial projection operator complex and limit the flexibility of this approach.

In (Thibaudeau *et al*., 2011) a similar strategy is used for 3D X-ray CT and object is discretized on a regular cylindrical grid with sector angles equal to the consecutive projection angles. This tessellation scheme has been shown to provide a partial projection operator limited to one projection angle and so it is found efficient. Moreover the simple tessellation scheme used in this models makes its implementation easily possible and so practically usable. Despite the advantages of this type of image representation, there are several questions on the performance of this method. These include: possibility for parallel implementation of projection and backprojection operator, use of penalty function in cylindrical coordinate and quality of reconstruction and reliability of it use for clinical imaging. Moreover, the variable pixel sizes used in this type of discretization affects the conditioning of problem and the convergence of numerical methods. These questions need to be studied and answered carefully before polar discretization of the object can be used for practical cases.

2.6 Conclusion

The reconstruction problem of the X-ray tomography can be represented by a linear deterministic model with additive noise. There are two approaches for solving this problem: analytical and statistical. Projection data from tomogrpahy systems are usually noisy. In analytical methods a simplified model is used for reconstruction and so the noise is not modeled. Consequently, these methods suffer from presence of noise in reconstructed images.

Statistical methods have shown to outperform analytical methods by providing higher quality image reconstruction. These methods provide high quality images by incorporating some prior knowledge on image and accounting for noise model. The objective function can then be represented as a Gaussian model with a weighted least squared term. This problem is ill-posed and adding a regularization function will guarantee that a unique solution exists. Different regularization functions have been used in the literature among which L_2 and L_2L_1 are most widely used. L_2 is of great interest because of its simple structure and L_2L_1 is used for its efficiency in preserving sharp edges.

Statistical methods can also be used to account for the polychromatic nature of the beam and

reduce metal artifacts. Use of these methods is hindered by their long reconstruction time and memory requirement and require further improvement before they can be used practically. Consequently, analytical methods are still the most widely used techniques in image reconstruction.

Different numerical methods have been introduced and used for tomography reconstruction. These methods mainly lie in two groups: gradient-based and quasi-Newton family. L-BFGS-B from the quasi-Newton family has shown interesting performance. This method has a super-linear convergence rate, a low memory need and can be used for solving non-quadratic optimization problems. Moreover, it can be easily used with preconditioners to improve the convergence.

The conventional method used for object representation is discretization on a regular Cartesian grid. In this method projection operator needs to be calculated at every projection angle. Storing this matrix requires a large amount of memory and is not possible for real size data. Consequently, the projection operator is usually calculated on-the-fly at every projection angle which makes the reconstruction time of statistical method long. Moreover, design of preconditioners for this type of representation is not trivial.

Using sector-invariant method for object representation allows benefiting from maximum symmetry in the system. This approach allows pre-calculation and storage of the partial projection operator with a limited memory requirement. The projection operation can then be performed by applying the projection operator by a simple circular shift of the object. Considering the limited size of the partial projection operator and the fact that projection matrix remains the same for all projections angles, design of preconditioners may be easier in this method. However, there are several open questions on the performance of this method and needs to be carefully studied. Some of these questions will be studied and addressed through this work.

CHAPTER 3

PROBLEM STATEMENT

Recall from chapter 2 that the linear forward model of tomography problems can be represented as Eq.(2.6). As already discussed, projection measurements in CT are noisy and a high quality reconstruction requires use of a model which accounts for this parameter. This justifies the use of statistical methods for image reconstruction; this was discussed in detail in chapter 2. The main drawback of current statistical methods for tomography reconstruction is their high memory need and the long reconstruction time.

In statistical method, an image is reconstructed iteratively using estimation algorithms. This is done by minimizing an objective function. MAP estimation is to be used due to its efficiency for solving ill-posed problems. The objective function which needs to be solved for this purpose was defined in (2.9):

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\geq 0}{\operatorname{argmin}_{\boldsymbol{\mu}}} \ \Phi(\boldsymbol{\mu}) \quad , \quad \Phi(\boldsymbol{\mu}) \triangleq \frac{1}{2}\|\mathbf{y}-\mathbf{P}\boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}_b} + \lambda R(\boldsymbol{\mu})$$

The objective function consists of two terms: a weighted least square term and a regularization function. The weighted least square term accounts for the noise model and the regularization function is used to incorporate some prior knowledge on the image. By adjusting the regularization parameter (λ) appropriately, one can obtain a good balance between resolution and noise level in the reconstructed image and control the trade-off between fidelity of solution to measured data and prior information.

For solving the above estimation problem, a few elements need to be decided on. These points can be summarized as below:

1. Object representation and system matrix
2. Regularization function
3. Noise model
4. Numerical method

In the subsequent sections these points will be discussed and the choices which will be made will be justified.

3.1 Object Representation

In Chapter 2 it was seen that when the object is discretized in Cartesian coordinates, pre-calcultion and storage of the projection operator becomes impossible for real size clinical data. The alternative is to calculate the projection operation on-the-fly for each projection angle. This results in long reconstruction time and loss of computational efficiency.

In contrast, it was seen that discretizing the object in polar coordinates and using the sector-invariant methods from the rotation-based approaches allow pre-calculation and storage of the partial projection operator. Using this method, reduces the size of projection matrix by as many times as the number of projection angles which leads to a dramatic reduction in memory need for X-ray CT imaging. This method is discussed in details in section 2.5.3.

As already discussed, using this type of representation for the object raises several questions. These points are summarized below:

- Discretizing the object in polar coordinates will result in having pixels with variable sizes. Therefore this type of representation is expected to affect the condition number of the problem significantly and reduce the convergence speed of numerical methods.
- It is not clear how the penalty function can be used in this model. This is because using a similar approach as the standard approach will further affect the condition number[1].
- Efficient implementation of the operations which are repeated at every iteration is another point which needs to be explored.

Once the effect of this type of discretization on the conditioning of problem is studied, the discussion on preconditioning in chapter 2 can be used in attempting to correct this problem.

Performance of this approach is not fully evaluated for X-ray CT imaging to our knowledge and is to be studied in this work. As these are difficult questions to be addressed and this is a preliminary study on the improvement of using polar coordinates for discretizing the object, all the developments will be done in a 2D framework for the sake of simplicity; extension to 3D will be possible and is left as future work. The parameters of the objective function will be chosen such that the objective function reduces to a simple model. This allow to fully explore the problems associated with this type of discretization.

3.2 Regularization function

In this work, quadratic regularization will be used. This will allow benefiting from the simple structure of this function and leads to a linear optimization problem to be solved. Using this quadratic

[1] This point will be discussed in detail in chapter 4

function, the estimation problem reduces to:

$$\hat{\boldsymbol{\mu}} = \underset{\substack{\boldsymbol{\mu} \\ \boldsymbol{\mu} \geq 0}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{P}\boldsymbol{\mu}\|^2_{\bar{\boldsymbol{\Sigma}}_b} + \lambda \|\mathbf{D}\boldsymbol{\mu}\|^2 \tag{3.1}$$

The solution to the optimization problem when the regularization function is quadratic has a closed-form as shown in (2.14) if the non-negative constraint is neglected. In solving this estimation problem iteratively projection and backprojection operations need to be performed repeatedly. Consequently a considerable amount of reconstruction time corresponds to performing these operations. Efficient implementation of these operations define the reconstruction time and its usability for clinical use. Therefore, possibility for parallelization needs to be considered when developing an algorithm.

The close-form solution of (2.14) can be used when analyzing the performance of the developed algorithm and comparing the quality of reconstruction using numerical data.

3.3 Noise model

The noise model which will be used in this work is Gaussian with constant variance. This reduces the complexity of problem and will result in a shift-invariant model for estimation problem. This property will be used in chapter 5 for designing preconditioners.

3.4 Numerical method

For solving the objective function iteratively, L-BFGS-B from the quasi-Newton family will be used. As discussed in chapter 2, this method has a super-linear convergence rate, low memory needs and can be used for solving non-quadratic optimization problems. Moreover, this method allows bound constraint with a decent convergence speed which is a desired property in image reconstruction. This method can also be used easily with preconditioners to improve the convergence.

3.5 Objectives

To summarize, in this work I will try to explore and answer the following questions on usability of polar coordinates for discretization of the object:

1. Computational efficiency of projection and backprojection and the possibility to parallelize both of these operations to reduce the reconstruction time of statistical methods,
2. Effect of variable pixel size on the conditioning of the problem and the speed of convergence,
3. Derivation of the penalty function when the object is discretized on a polar grid and,
4. Actual quality of reconstructed images.

CHAPTER 4

OBJECT DISCRETIZATION IN POLAR COORDINATES

In Section 2.5, it was seen that representing the object on a polar grid (cylindrical grid for 3D), allows taking advantage of the maximum symmetries in the system and scanner geometry and reduces the memory requirement considerably. This is because for this type of object representation one needs to calculate the partial projection operator for one reference angle and then rotate the object at every projection angle. Consequently, the number of non-redundant elements which need to be stored decreases considerably.

In chapter 3, the problems associated with this type of representation were discussed and the framework which will be used in this work was justified.

I will try to address these problems in this chapter and the next two chapters. This includes questions about how the penalty function should be used in this type of representation and how the conditioning of the problem is affected (to be addressed in chapters 4 and 5). Actual quality of the reconstructed images will be studied experimentally through simulations in chapter 6, where results are provided.

4.1 Image representation in polar coordinates

As discussed in Section 2.3.2, in image reconstruction using iterative methods, an object is discretized into some pixels and the attenuation map of the object is calculated using estimation methods. Standard approaches discretize the object in Cartesian coordinates. In this work, polar coordinates will be used for this purpose. The tessellation scheme which will be used for this purpose needs to have two properties: first, it needs to preserves the symmetry properties of system, and second should be easily implementable. The symmetric property of the problem is preserved if angular samples are equally spaced.Moreover, the selected tesselation scheme should allow precise representation of the object in order for the method to be useful in a clinical setting. This means pixellation scheme with holes as in (Mora *et al.*, 2008) with circular pixels, is not suitable.

Here, it is assumed that the object is discretized onto a regular polar grid with the number of angular samples equal to the number of regularly spaced projections. The algorithm which will be developed here can be easily extended to cases where the radial samples are not equally spaced or the angular samples are an integer multiple of the number of projection angles. Fig.4.1 shows the pixellation scheme used in this work.

The discretized object in this type of discretization can be represented as $\boldsymbol{\mu}(r,\theta)$. After $\boldsymbol{\mu}(r,\theta)$

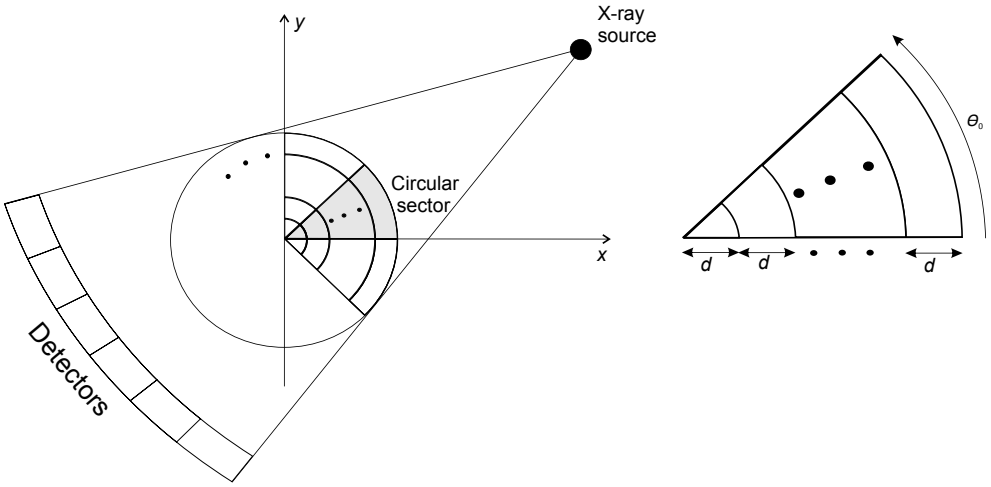


Figure 4.1 Discretization of the object in polar coordinates.

has been estimated, the attenuation map can be easily converted to Cartesian coordinates using following relationship:

$$\boldsymbol{\mu}(\vec{x}) = \mathbf{T}_{pc}\boldsymbol{\mu}(r,\theta)$$

$\mathbf{T}_{pc}$ in above relationship denotes a polar-to-Cartesian transformation which is basically an interpolation.

4.2 Assumptions and reconstruction framework

The linear forward model derived in (2.6) will be used for a 2D axial CT reconstruction problem with additive noise modeled as **b**:

$$\mathbf{y} = \mathbf{P}\boldsymbol{\mu} + \mathbf{b}$$

where $\mathbf{y} \in \mathbb{R}^{MN}$, $\boldsymbol{\mu} \in \mathbb{R}^{KN}$ and $\mathbf{b} \in \mathbb{R}^{MN}$ respectively denote the sinogram, attenuation map of the object and the measurement noise. The system matrix is represented by **P** and has a block-circulant form (this will be discussed more in Section 4.4.1). Here N represents the number of angular samples, K is the number of radial samples of the object and M is the number of data samples per projection angle or the number of detectors at each instance of data acquisition. It is assumed that the samples in **y** and $\boldsymbol{\mu}$ are regularly arranged by increasing angle values.

For $\mathbf{b} \rightsquigarrow \mathcal{N}(0, \boldsymbol{\Sigma}_b^{-1})$ and using MAP estimation for reconstructing the image, the problem to be solved reduces to (2.9).

Matrix $\boldsymbol{\Sigma}_b$ is a diagonal matrix and represent the noise variance. Here, the noise is Gaussian with constant variance, i.e., this matrix is set to identity. $\lambda R(\boldsymbol{\mu})$ is the regularization term which applies a L_2 (quadratic) penalty on $\boldsymbol{\mu}$ itself and on its first differences on first-order neighborhoods.

4.3 System matrix computation

By using the polar coordinates for representing the object, with the tessellation scheme shown in Fig.(4.1), the object will be discretized in N equally spaced sectors. Consequently, this representation introduces symmetry and results in having a block-circulant projection matrix[1]. This introduces redundancy in the projection operator. Storing only the non-redundant elements of this matrix, reduces the memory requirement by a factor equal to N, compared to the standard Cartesian grid.

It should be noted that, as discussed in section 2.5.3, using this type of representation requires no interpolation during the estimation process and so does not suffer from the interpolation errors which may exist in other rotation-based methods. Moreover, the large number of redundancies in the projection operator makes pre-calculation of the partial projection operator possible and non-redundant parameters can easily fit in the random access memory (RAM) of any machine.

Calculation of the coefficients of the projection operator can be done using simple geometrical equations. Different methods are available for calculating these coefficients, see AppendixA. In this work, ray-tracing techniques will be used. The effect of ray thickness in practical problems may also be accounted for by considering several thin rays per detector.

Remark 4.3: The projection operator for 3D image reconstruction can be similarly calculated using cylindrical coordinates as in (Leroux *et al.*, 2007).

4.4 Projection and backprojection

As already discussed, projection and backprojection are performed at every iteration. A significant amount of reconstruction time corresponds to these operations and so efficient implementation of them can reduce the long reconstruction time of statistical methods.

Researchers have attempted to achieve this goal by parallel implementation of these operations. Although some interesting results have been achieved for the projection operator, parallel implementation of both projection and backprojection is still known to be challenging, because both row and column access to $\mathbf{P}$ is required.

Parallel implementation of projection and backprojection becomes crucial in 3D image reconstruction. In this section the possibility of parallel implementation of both projection and backprojection operations with polar grid representation of the object will be discussed.

[1] This property will be further discussed when deriving the efficient implementation of projection and backprojection operations in section 4.4

4.4.1 Projection

Let $\mathbf{y}_n$ denote the projection measurements for projection angles θ_n; $0 \leq n \leq N-1$, and $\mathbf{P}_0$ denote the projection matrix for reference angle θ_0. One can pre-compute and store this partial projection matrix as explained in Section 4.3.

The complete projection is obtained by rotating the object by $-\theta_n$ for each projection angle and then applying $\mathbf{P}_0$ to get $\mathbf{y}_n$. Rotation of the object at every projection angle can be performed by using a K-sample up circular shift operator. Note that these rotations involve no arithmetic computation and can be performed using index manipulation.

Now, for implementing the projection operator, let us derive the relationship between the partial projection operator $\mathbf{P}_0$ discussed above and the complete projection operator $\mathbf{P}$. If $\mathbf{S}_\mu^{-1}$ represents the K-sample circular shift operator, this matrix will be applied to the vector of object $\boldsymbol{\mu}$, n times to obtain the complete projection operator. The complete projection operator can then be expressed by following relation (Goussard *et al*., 2013):

$$\mathbf{P} = \left[\mathbf{P}_0^T \mid (\mathbf{P}_0\mathbf{S}_\mu^{-1})^T \mid \ldots \mid (\mathbf{P}_0\mathbf{S}_\mu^{-N+1})^T\right]^T \tag{4.1}$$

Now, looking at (4.1) we can see that projection matrix $\mathbf{P}$ has a (N,N) block-circulant structure. The partial projection matrix corresponding to the reference angle can be represented as following:

$$\mathbf{P}_0 = [\mathbf{P}_{0,0} \mid \mathbf{P}_{0,1} \mid \ldots \mid \mathbf{P}_{0,N-1}] \tag{4.2}$$

where each $\mathbf{P}_{0,n}$; $0 \leq n \leq N-1$ is a (M,K) block. Substituting (4.2) in (4.1), the complete projection matrix can be represented as following which is equal to (4.1):

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \ldots & \mathbf{P}_{0,N-1} \\ \mathbf{P}_{0,N-1} & \mathbf{P}_{0,0} & \ldots & \mathbf{P}_{0,N-2} \\ \vdots & & \ddots & \\ \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \ldots & \mathbf{P}_{0,0} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_0\mathbf{S}_\mu^{-1} \\ \vdots \\ \mathbf{P}_0\mathbf{S}_\mu^{-N+1} \end{bmatrix} \tag{4.3}$$

This block-circulant property of the projection matrix makes efficient implementation of projection and backprojection possible. Indeed, looking at the block-row decomposition of projection operator in 4.3, one can see that the projections $(\mathbf{y}_n)$ at different angles can be calculated independently from one another. This is because having the partial projection matrix, one can apply the $\mathbf{S}_\mu^{-1}$ operator to any vector through index manipulations and calculate the corresponding projection. This operation can be performed in parallel for all projection angles with no arithmetic computation for rotations. This shows that the special structure of $\mathbf{P}$ easily allows parallel implementation of the projection operation.

Products of the projection operator by the vector of object can be achieved efficiently using available public libraries. Moreover, the computational structure of this operation makes general

purpose parallelization libraries such as OpenMP® easy to be used.

4.4.2 Backprojection

The shift matrix $\mathbf{S}_\mu$ has an orthogonal property, i.e.: $\mathbf{S}_\mu^T = \mathbf{S}_\mu^{-1}$. Therefore, backprojection of the projection measurements for projection angle θ_n ($\mathbf{y}_n$), onto a vector $\boldsymbol{\mu}^{(b)}$ can be represented by following relationship (Goussard *et al.*, 2013):

$$\boldsymbol{\mu}^{(b)} = \sum_{n=0}^{N-1} \mathbf{S}_\mu^n \mathbf{P}_0^T \mathbf{y}_n$$

Implementation of backprojection using this method is easy but does not allow parallel implementation. This is where the block-circulant structure of $\mathbf{P}$ becomes important. One can address this problem, by partitioning $\mathbf{P}$ in a block-column manner:

$$\mathbf{P} = \left[\mathbf{Q}_0 \mid \mathbf{S}_y^{-1}\mathbf{Q}_0 \mid \ldots \mid \mathbf{S}_y^{-N+1}\mathbf{Q}_0\right], \tag{4.4}$$

Where,

$$\mathbf{Q}_0 = \left[\mathbf{P}_{0,0}^T \mid \mathbf{P}_{0,N-1}^T \mid \mathbf{P}_{0,N-2}^T \mid \ldots \mid \mathbf{P}_{0,1}^T\right]^T \tag{4.5}$$

Here $\mathbf{S}_y$ is the M-sample up circular shift operator. (4.5) can be directly written from the projection operator structure given in (4.3). $\mathbf{P}^T$ will then have the following form:

$$\mathbf{P}^T = \begin{bmatrix} \mathbf{Q}_0^T \\ (\mathbf{S}_y^{-1}\mathbf{Q}_0)^T \\ \vdots \\ (\mathbf{S}_y^{-N+1}\mathbf{Q}_0)^T \end{bmatrix} \tag{4.6}$$

using $\mathbf{S}^T = \mathbf{S}^{-1}$ for orthogonal matrices, we will then have:

$$\mathbf{P}^T = \begin{bmatrix} \mathbf{Q}_0^T \\ \mathbf{Q}_0^T\mathbf{S}_y \\ \vdots \\ \mathbf{Q}_0^T\mathbf{S}_y^{N-1} \end{bmatrix} \tag{4.7}$$

We can see in (4.7) that $\mathbf{P}^T$ has a similar form as $\mathbf{P}$ in (4.1). Therefore, $\mathbf{P}^T$ can be decomposed in a same manner and efficient implementation of backparojection can be achieved. Similar to parallel implementation of projection, $\mathbf{S}_y^n$ operations here can be applied to projection measurments ($\mathbf{y}_n$) independently to obtain $\boldsymbol{\mu}$. This means that the loop over different projection angles for calculating the product of $\mathbf{S}_y^n\mathbf{y}_n$ with the corresponding part of the $\mathbf{P}^T$ can be done independently and can be

parallelized to reduce the reconstruction time.

This discussion shows that using polar-coordinates for discretizing the object leads to efficient implementation of both projection and backprojection operations in parallel.

4.5 Penalty function in polar coordinates

The necessity of adding a penalty function to objective function was discussed in Chapter 2. As explained in Section 2.3.2.2, regularization function $R(\mu)$ usually applies a penalty on the first differences of $\boldsymbol{\mu}$. This penalty is based on a numerical approximation to the gradient of μ. Approximation to the gradient is proportional to the distances between neighboring pixels, which in a Cartesian grid are identical for all pixels. For polar-grid discretization, this is not the case and distances between neighboring pixels vary significantly along the tangential direction. Therefore, it is not clear how this penalty term should be applied for this type of representation of the object.

When the object is discretized on a polar grid, it is mathematically rational to penalize the neighboring pixels based on the inverse of the distance between pixels to be consistent with the definition of the gradient and the physical meaning of the penalty term. This will be refereed to as nonuniform weighting in this work. However, the problem with this approach is that weighting the pixels in this manner gives a very high weight to pixels located in the center of image and consequently affects the condition number of problem. Therefore, it is not clear how this will affect the convergence of estimation problem and the quality of the reconstructed images. Another possibility is to weight the first differences in a uniform manner similar to weightings used when the object is discretized in cartesian coordinates. This penalty function will treat the pixels equally independently of their sizes and so does not affect the conditioning of problem.

Here, both of these approaches will be tested to assess both the quality of reconstruction and the convergence speed.

The conditioning of the problem for both type of regularization will be studied in next Section 4.6.

The regularization term $\lambda R(\mu)$, can be written as:

$$\lambda R(\boldsymbol{\mu}) = \lambda \sum_{m=1}^{\bar{M}} \sum_{i=1}^{KN} \psi_i([\mathbf{D}^{(m)}\boldsymbol{\mu}]_i) \tag{4.8}$$

where $\mathbf{D}$, first difference matrix, is $KN \times KN$ and ψ is the regularization function and here is chosen to be L_2: $\psi(\boldsymbol{\mu}) = \boldsymbol{\Omega}\frac{\mu^2}{2}$, with $\boldsymbol{\Omega}$ being the weighting matrix. Index m, corresponds to the neighboring direction which will be considered and is defined as follows:

- $m = 1$, radial differences,
- $m = 2$, tangential differences,

$\bar{M}$ is the total number of neighboring direction which will be considered in the penalty term[1]. This gives:

$$\lambda R(\boldsymbol{\mu}) = \lambda \sum_{m=1}^{\bar{M}} \sum_{i=1}^{KN} \psi_i([\mathbf{D}^{(m)}\boldsymbol{\mu}]_i) = \frac{\lambda}{2} \sum_{m=1}^{\bar{M}} \boldsymbol{\mu}^T \mathbf{D}^{(m)^T} (\Omega^{(m)})^2 \mathbf{D}^{(m)} \boldsymbol{\mu} \tag{4.9}$$

Matrices of differences ($\mathbf{D}^{(m)}$) are sparse and have block-circulant structures for both type of regularizations discussed below and are presented in Appendix C. This property will be used in Chapter 5 for designing preconditioners.

4.5.1 Uniform weighting

Assuming that uniform weighting is used for the regularization function, weighting matrices $\boldsymbol{\Omega}^{(m)}$ will be equal to identity for $m \in \{1,2\}$. Here, we are only considering the radial and tangential neighbors ($\bar{M} = 2$) for simplicity and fair comparison with nonuniform weighting which will be discussed next.

4.5.2 Nonuniform weighting

In this work, the object is discretized such that distances between pixels are identical in the radial direction.

$$\boldsymbol{\Omega}^{(1)} = \frac{1}{d} \times \mathbf{I}(KN, KN) \tag{4.10}$$

Here, $\mathbf{I}$ is the identity matrix and d is the difference between two consecutive radiuses in polar coordinates. The distance between pixels varies significantly in the tangential direction. $\boldsymbol{\Omega}^{(2)}$ will have the form given in (4.11). Each block of this matrix is applied to one sector of the discretized object and the coefficients which are applied to each pixel correspond to the inverse of the distance of this pixel to its neighboring pixel in the tangential direction.

[1] $m = 3$ and 4 can be also added to include the diagonal neighboring pixels in south-east and north-east directions as well. Matrices of differences for these directions are given in Appendix C

$$\mathbf{\Omega}^{(2)} = \frac{N}{2\pi d} \times \begin{bmatrix} \begin{bmatrix} \frac{1}{1-\frac{1}{2}} & & & \\ & \frac{1}{2-\frac{1}{2}} & & \\ & & \ddots & \\ & & & \frac{1}{K-\frac{1}{2}} \end{bmatrix}_{K\times K} & & 0 \\ & \ddots & \\ 0 & & \begin{bmatrix} \frac{1}{1-\frac{1}{2}} & & & \\ & \frac{1}{2-\frac{1}{2}} & & \\ & & \ddots & \\ & & & \frac{1}{K-\frac{1}{2}} \end{bmatrix}_{K\times K} \end{bmatrix} \tag{4.11}$$

4.6 Conditioning

Discretizing object in polar coordinates results in having pixels with variable size. This may affect the condition number of the problem and so make the convergence slower. The definition of the condition number and its impact on the speed of convergence for iterative methods was discussed in Section 2.4.2.

To compare how the condition number is affected in polar-discretization, the eigenvalue spectrum of the normal matrix when object is discretized in both Cartesian coordinates and polar coordinates is calculated for a typical case. Here the impact of the regularization function is omitted by using λI for penalty function. This reduces the normal matrix to $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$. Fig.(4.2), shows how the eigenvalues are spread in both cases. One can see in this figure that, in the polar case, eigenvalues are spread on a wider range compared to the Cartesian case and so more iterations and consequently longer reconstruction time is expected.

The problem of conditioning becomes more severe when the regularization function with nonuniform weighting is used (See Section 4.5). To see the effect of the regularization function, the eigenvalue spectrum of the normal matrix for each type is plotted in Fig.(4.3). The value of λ is chosen to be high (10) to highlight the effect of nonuniform weighting on conditioning of problem. From this figure we can see that using nonuniform weighting conditioning is deteriorated more.

The conditioning of the problem needs to be improved for this method to be useful. For this purpose, one may attempt to use preconditioners such as those discussed in section 2.4.2.

In the next chapter which is specifically devoted to preconditionig we will attempt to design different types of preconditioners. The performance and efficiency of each preconditioner will then be explored in Chapter 6.

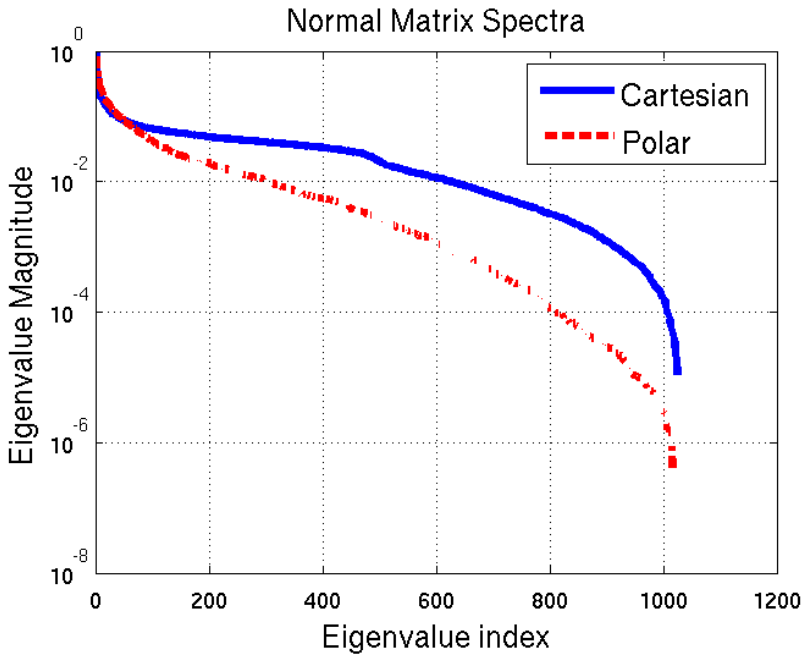


Figure 4.2 Eigenvalue spectrum of the normal matrix $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I}$ in the Cartesian and polar representations for $\lambda = 10^{-3}$.

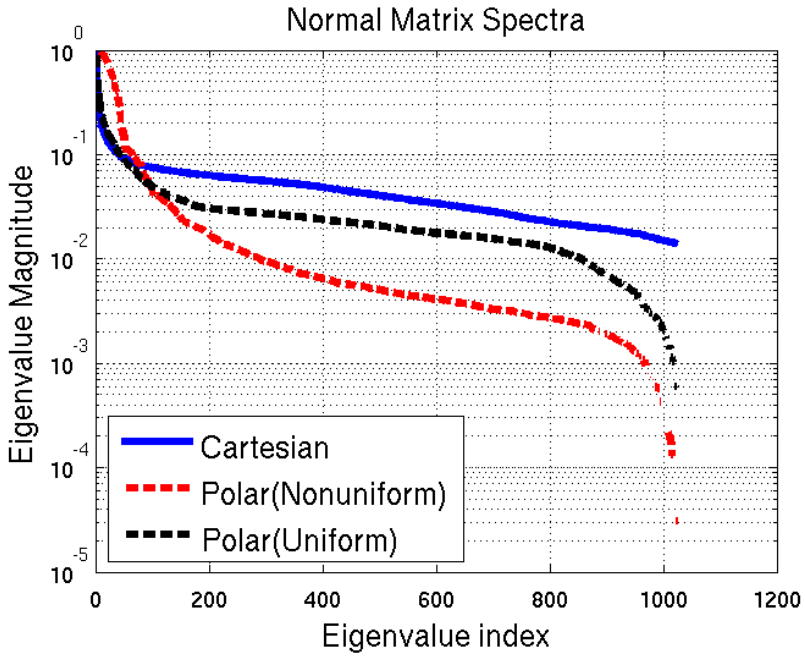


Figure 4.3 Eigenvalue spectrum of the normal matrix $\mathbf{P}^T\mathbf{P} + \lambda\mathbf{D}^T\mathbf{D}$ in the Cartesian and polar representations for $\lambda = 10$.

CHAPTER 5

PRECONDITIONING

In chapter 4 it was seen that using a polar grid for discretizing the object introduces variable pixel sizes and affects the conditioning of problem and so the convergence of numerical methods. Moreover, in section 4.6 it was seen that this problem becomes more severe when the nonuniform regularization is used for penalty function.

In any case, deterioration of the conditioning caused by polar discretization of the object needs to be corrected if this method is to be used in clinical scanners. In section 2.4.2, the definition of the condition number and how it affects the convergence was discussed. Moreover, it was seen that, using efficient preconditioners conditioning of the problem can be improved and the convergence speed can be increased.

In this chapter we will first study the normal matrix structure when the polar grid is used for discretizing the object. We will then explore the different types of preconditioners which can be designed by taking advantage of this structure. The procedure for designing each preconditioner as well as its implementation will be provided in subsequent sections and, the approximations involved in each case will be discussed.

5.1 Normal matrix structure

In section(2.4.2) it was seen that, in solving an optimization problem using estimation techniques, the convergence speed is dependent on the conditioning of the Hessian of the objective function. Hessian is referred to as the normal matrix here. For a quadratic objective function as in this work (3.1), this matrix is independent of the variables at which the optimization problem is to be solved for, and has the following form:

$$\nabla^2\Phi(\boldsymbol{\mu}) = \mathbf{P}^T\mathbf{P} + \lambda\{\mathbf{D}^{(1)^T}(\boldsymbol{\Omega}^{(1)})^2\mathbf{D}^{(1)} + \mathbf{D}^{(2)^T}(\boldsymbol{\Omega}^{(2)})^2\mathbf{D}^{(2)}\} \tag{5.1}$$

and is positive-definite and Hermitian.

The matrices of differences ($\mathbf{D}^{(m)}$) are known to be block-circulant (BC)[1] and consequently, their transpose is also BC. From the properties of BC matrices, multiplication of two BC matrices will be also BC, resulting in $\mathbf{D}^{(\mathbf{m})^T}\mathbf{D}^{(m)}$ being BC. Now, considering the BC structure of the projection matrix $\mathbf{P}$ demonstrated in (4.3), $\mathbf{P}^T\mathbf{P}$ will also have a BC form:

[1] Refer to section 4.5.

$$\mathbf{P}^T\mathbf{P} = \begin{bmatrix} \mathbf{Q}_0^T \\ \mathbf{Q}_0^T\mathbf{S}^1 \\ \vdots \\ \mathbf{Q}_0^T\mathbf{S}^{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_0 & \mathbf{S}^{-1}\mathbf{Q}_0 & \dots & \mathbf{S}^{-N+1}\mathbf{Q}_0 \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{Q}_0^T\mathbf{Q}_0 & \mathbf{Q}_0^T\mathbf{S}^{-1}\mathbf{Q}_0 & \dots & \mathbf{Q}_0^T\mathbf{S}^{-N+1}\mathbf{Q}_0 \\ \mathbf{Q}_0^T\mathbf{S}^1\mathbf{Q}_0 & \mathbf{Q}_0^T\mathbf{Q}_0 & \dots & \mathbf{Q}_0^T\mathbf{S}^{-N+2}\mathbf{Q}_0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_0^T\mathbf{S}^{N-1}\mathbf{Q}_0 & \mathbf{Q}_0^T\mathbf{S}^{N-2}\mathbf{Q}_0 & \dots & \mathbf{Q}_0^T\mathbf{Q}_0 \end{bmatrix} \tag{5.2}$$

Consequently, the normal matrix which has a strong impact on the convergence of the optimization algorithm, has a BC form. Note that this property is valid when $\mathbf{\Sigma} = \mathbf{I}$. This property will be used in designing preconditioners in this chapter.

Before starting to design preconditioners and exploring their efficiency, let us have a closer look at the normal matrix structure. The normal matrix consists of the projection matrix and the matrices of differences. The matrices of differences are block-diagonal or near block-diagonal with dominant entries on the main diagonal (Appendix C). For the projection matrix, in (5.2), we can see that the block-diagonal elements of $\mathbf{P}^T\mathbf{P}$ are identical and are equal to $\mathbf{Q}_0^T\mathbf{Q}_0$. Fig.5.1 shows the $\mathbf{P}^T\mathbf{P}$ in an image form, for a small size of data. This figure suggests that the dominant values

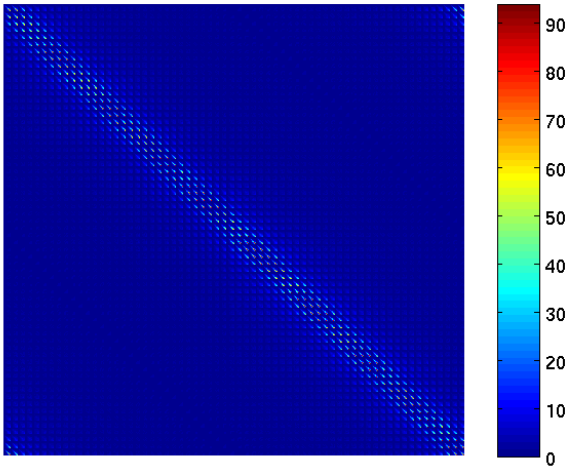


Figure 5.1 Illustration of $\mathbf{P}^T\mathbf{P}$ in image form, for a small size of image (32×32).

are located on the diagonal of this matrix which consists of these identical blocks ($\mathbf{Q}_0^T\mathbf{Q}_0$). To further investigate this point, the L_2 norm of the N blocks of size K of the partial projection matrix corresponding to the reference angle is calculated. One can see in Fig.(5.2) that the first block

which corresponds to $\mathbf{Q}_0^T\mathbf{Q}_0$, has the largest norm.

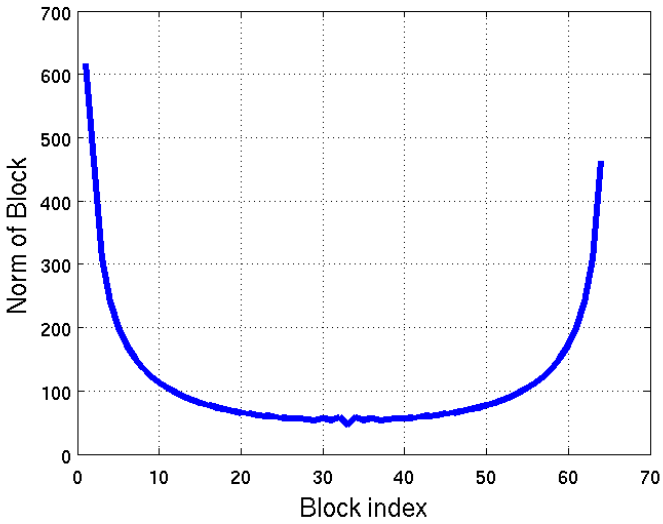


Figure 5.2 $\|\mathbf{P}^T\mathbf{P}\|$ for the reference angle.

In Fig.(5.1) we can also see that the off-diagonal elements of $\mathbf{P}^T\mathbf{P}$ are very close to zero and so sparse approximation to this matrix does not introduce a large inaccuracy.

To explore the properties of the block $\mathbf{Q}_0^T\mathbf{Q}_0$, this matrix is depicted in Fig.(5.3) for a small size of the object. One can see that this block has a similar property as $\mathbf{P}^T\mathbf{P}$: the dominant values are located on the diagonal and the off-diagonal elements are close to zero. Therefore, we can easily make a sparse approximation to this block without a large lose of accuracy.

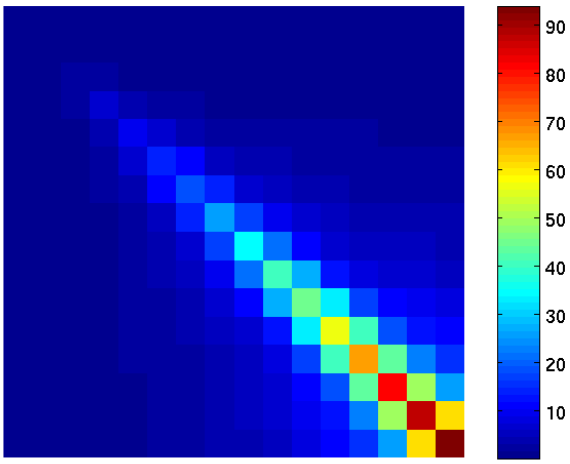


Figure 5.3 Illustration of $\mathbf{Q}_0^T\mathbf{Q}_0$ in image form, for a small size of image (32×32).

Next section discusses how these properties will be used for designing preconditioners.

5.2 Preconditioner design

In designing preconditioners, we are trying to find a good trade-off between a good approximation to the inverse of the normal matrix and the memory need. Having a closer approximation will guarantee the convergence in a smaller number of iterations. The structure of normal matrix was discussed in the previous section, in this chapter we will try to find an approximation to normal matrix that lends itself to easy factoring and inversion. This matrix can then be used as preconditioner to improve the convergence.

Considering the special structure of normal matrix and its properties illustrated in the previous section, one can think of different types of preconditioners. It was seen in Fig.(5.1) that the magnitude of the coefficients decreases when moving away from the diagonal with off-diagonal elements being very close to zero. This suggests that one good approximation to the normal matrix is to use the diagonal or block-diagonal elements of this matrix for designing the preconditioner.

Another approach for designing the preconditioner is to use the fact that a BC matrix can be block-diagonalized in a fast way using FFT techniques. However, it should be noted that although using the block-diagonalized form of the normal matrix decreases the memory need significantly ($K^2 \times N$ as opposed to $K^2 \times N^2$) this matrix is still large for practical applications. This is because the blocks are not identical. Therefore, some diagonal or sparse approximation to this block-diagonalized normal matrix has to be used for designing preconditioners.

Consequently, for either of these approaches, two type of preconditioners can be designed. The preconditioners will be either diagonal or a sparse block-diagonal matrix. The main advantage of diagonal preconditioners is their low memory need for storage and the negligible cost for computations per iteration. However, the improvement of convergence by these type of preconditioners is usually limited. Non-diagonal preconditioners may provide a better efficiency but at the cost of more memory need and more computations involved.

When using preconditioners, one main concern is how to invert the preconditioning matrix. For diagonal preconditioners with non-zero entry on the diagonal this is not an issue, as the inverse of a diagonal matrix is diagonal with each diagonal element inverted. For non-diagonal preconditioners, inversion of the preconditioning matrix may be omitted if the corresponding linear system of equations can be solved efficiently at each iteration.

Below, the preconditioners which will be explored in this chapter are listed and categorized:

1. Approach 1: Block diagonal approximation to the normal matrix
   (a) Diagonal Preconditioners

- Preconditioning based on pixel size,
- Preconditioning based on the inverse of the diagonal elements of the normal matrix,
- Preconditioning based on the diagonal elements of the inverse of an approximation to the normal matrix,

(b) Cholesky decomposition

- Complete Cholesky decomposition of an approximation to the normal matrix,
- Incomplete Cholesky decomposition of an approximation to the normal matrix,

2. Approach 2: Block-diagonalization of the normal matrix using FFT

(a) Diagonal preconditioning,

(b) Incomplete Cholesky decomposition pf a sparse approximation to the block-diagonalized normal matrix.

The reasons for selecting these preconditioners and their respective performance will be discussed and explored in subsequent sections.

Remark 5.1: Here, in the list of preconditioners which will be designed using first approach, a diagonal preconditioner based on pixel size is also included. This preconditioner is designed by making a heuristic approximation to $\mathbf{P}^T\mathbf{P}$ and considering the variable pixel size of object discretized in polar-coordinates. The efficiency of this preconditioner will be tested experimentally in the next chapter.

Remark 5.2: As mentioned in chapter 3, the numerical method which will be used for image reconstruction in chapter 6 is L-BFGS-B. In this numerical method, preconditioner can be applied using a change of variable. According to Remark 2.4.2.1, preconditioning then requires that the Hessian be left and right multiplied by two matrices. This point will be considered in designing preconditioners in this chapter.

5.3 Approach 1: Block diagonal approximation to the normal matrix

According to the above discussion and considering that the dominant blocks of the normal matrix are located on the main diagonal, one good approach is to make a block diagonal approximation to normal matrix. This approximation can be performed for the least-squares part and the regularization part separately without loss of generality.

For least squares part, the approximation to (5.2) has the following form:

$$\widetilde{\mathbf{P}^T\mathbf{P}} = \underbrace{\begin{bmatrix} \mathbf{\Lambda} & & & \\ & \mathbf{\Lambda} & & \\ & & \ddots & \\ & & & \mathbf{\Lambda} \end{bmatrix}}_{KN\times KN} \tag{5.3}$$

where $\mathbf{\Lambda} = \mathbf{Q}_0^T\mathbf{Q}_0$, and have size $K \times K$.

Similarly, a block-diagonal approximation to the matrices of first differences can be performed. In section 5.1 it was discussed that these matrices are very close to block-diagonal with dominant values located on the diagonal. The general structure of these matrices and their multiplication with their transpose is provided in Appendix C. The block-diagonal approximation to $\mathbf{D}^T\mathbf{\Omega}^T\mathbf{\Omega}\mathbf{D}$ matrices will then be:

$$\widetilde{\mathbf{D}^{(1)^T}\mathbf{\Omega}^{(1)^T}\mathbf{\Omega}^{(1)}\mathbf{D}^{(1)}} = \underbrace{\begin{bmatrix} \mathbf{\Delta}_r & & & \\ & \mathbf{\Delta}_r & & \\ & & \ddots & \\ & & & \mathbf{\Delta}_r \end{bmatrix}}_{KN\times KN}, \quad \widetilde{\mathbf{D}^{(2)^T}\mathbf{\Omega}^{(2)^T}\mathbf{\Omega}^{(2)}\mathbf{D}^{(2)}} = \underbrace{\begin{bmatrix} \mathbf{\Delta}_t & & & \\ & \mathbf{\Delta}_t & & \\ & & \ddots & \\ & & & \mathbf{\Delta}_t \end{bmatrix}}_{KN\times KN} \tag{5.4}$$

Matrices $\mathbf{\Delta}_r$ and $\mathbf{\Delta}_t$ are $K \times K$ and represent the corresponding blocks in $\mathbf{D}^T\mathbf{\Omega}^T\mathbf{\Omega}\mathbf{D}$ for penalizing the radial and tangential neighboring pixels, respectively.

Using the block-diagonal approximations made (5.3 and 5.4), the block-diagonal approximation to the Hessian of objective function $\Phi(\mu)$ will then have the following form:

$$\widetilde{\nabla^2_{\mu}\Phi(\boldsymbol{\mu})} = \begin{bmatrix} \mathbf{\Lambda} & & & \\ & \mathbf{\Lambda} & & \\ & & \ddots & \\ & & & \mathbf{\Lambda} \end{bmatrix} + \lambda \left(\begin{bmatrix} \mathbf{\Delta}_r & & & \\ & \mathbf{\Delta}_r & & \\ & & \ddots & \\ & & & \mathbf{\Delta}_r \end{bmatrix} + \begin{bmatrix} \mathbf{\Delta}_t & & & \\ & \mathbf{\Delta}_t & & \\ & & \ddots & \\ & & & \mathbf{\Delta}_t \end{bmatrix} \right) \tag{5.5}$$

Extension of the above results to the case where more neighboring pixels including the neighboring pixels in diagonal directions are considered in the penalty function is straightforward.

Considering the fact that the diagonal blocks in (5.5) are identical, and taking the first block of size $K \times K$, we will have:

$$\begin{aligned} \widetilde{\nabla^2_{\mu}\Phi_1(\boldsymbol{\mu})} &= \mathbf{\Lambda} + \lambda\{\mathbf{\Delta}_r + \mathbf{\Delta}_t\} \\ &= \mathbf{Q}_0^T\mathbf{Q}_0 + \lambda\mathbf{\Delta}_{\boldsymbol{tot}} = \mathbf{\Upsilon} \end{aligned} \tag{5.6}$$

Considering the structure of $\widetilde{\nabla^2\Phi_1}$ derived above with blocks $\boldsymbol{\Upsilon}$, here, we are trying to find a block diagonal preconditioner $\boldsymbol{\Pi}$ with the same block-diagonal structure as the approximated Hessian. The blocks of this matrix will be identical, each diagonal block being a full, sparse or diagonal approximation to $\boldsymbol{\Upsilon}$. In following subsections different preconditioners will be designed using $\boldsymbol{\Upsilon}$ derived in this section.

5.3.1 Diagonal Preconditioners

Diagonal preconditioners are the most simple type of preconditioners which can be used to correct the conditioning of the problem. These preconditioners are of interest because of their low memory need and negligible added calculation time per iteration. Applying these preconditioners to the objective function (2.20)is very economical as the left and right multiplication by a diagonal matrix is virtually costless. These preconditioners are also known to be easily implementable and so practically usable. In this section three different diagonal preconditioners are suggested to be tested for their efficiency.

5.3.1.1 Preconditioning based on pixel size

As discussed in section 2.4.2, discretizing image onto a regular polar grid yields variable pixel sizes. It is expected that if a diagonal preconditioner which accounts for this different pixel sizes is used, this deterioration of conditioning can be partly corrected.

This preconditioner will have following form:

$$\boldsymbol{\Pi}_1 = \left(\boldsymbol{\Gamma}^{(1)} + \lambda\, diag(\boldsymbol{\Delta_{tot}})\right)^{-\frac{1}{2}} \tag{5.7a}$$

where $\boldsymbol{\Delta}_{tot}$ is defined in (5.6), and

$$\boldsymbol{\Gamma}^{(1)} = \begin{bmatrix} \frac{1}{1-\frac{1}{2}} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{2-\frac{1}{2}} & 0 & \dots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \dots & \frac{1}{M-\frac{1}{2}} \end{bmatrix} \tag{5.7b}$$

Here the diagonal approximation to $\boldsymbol{\Upsilon}$ is the inverse of the pixel sizes in one sector. By doing this, we are reducing the high weight given to the small pixels located close to center of image in polar-discretization and make the normal matrix more uniform[1]. Moreover, as explained in Remark 5.2, the preconditioning matrix will be applied to normal matrix from both left and right sides, so the the square-root of this matrix needs to be used as preconditioner.

[1] Note the similarity between $\boldsymbol{\Gamma}^{(1)}$ and the blocks in $\boldsymbol{\Omega}^{(2)}$

5.3.1.2 Inverse of diagonal elements of normal matrix

In section 5.1 it was concluded that the dominant values of the normal matrix are located on the main diagonal. Therefore, a simple choice for diagonal preconditioning is to make a diagonal approximation to Υ in (5.6).

This diagonal preconditioner will then have the following form:

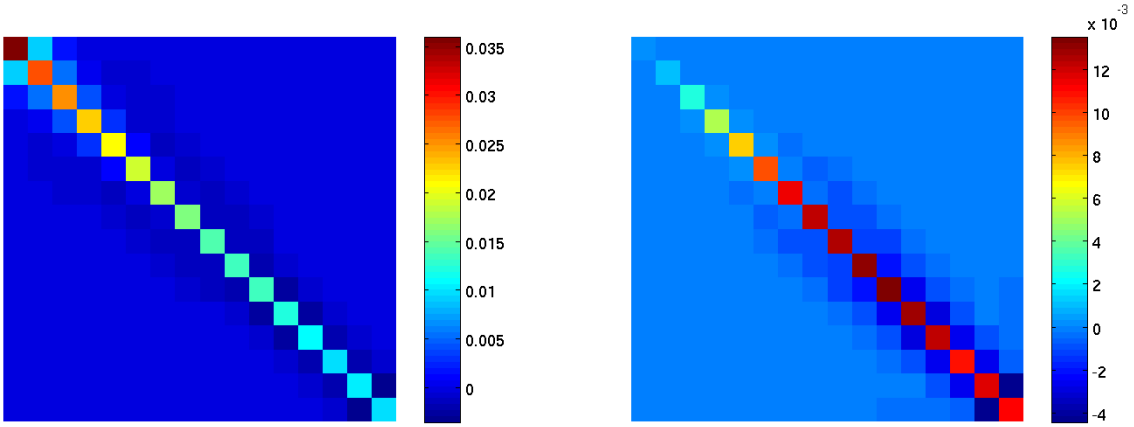
$$\mathbf{\Pi}_1 = \left(\mathbf{\Gamma}^{(2)} + \lambda diag(\mathbf{\Delta}_{tot})\right)^{-\frac{1}{2}} \tag{5.8a}$$

where

$$\mathbf{\Gamma}^{(2)} = diag(\mathbf{Q}_0^T \mathbf{Q}_0) \tag{5.8b}$$

5.3.1.3 Diagonal elements of the inverse of Υ

Another approach for designing diagonal preconditioner is to calculate the exact inverse of Υ and use the diagonal values of this matrix for preconditioning. Fig.(5.4a) and (5.4b) shows Υ^{-1} for both types of regularization in image form for a typical case. One can see from these figures that similarly to the previous case, the dominant values are located on the main diagonal.



(a) (b)

Figure 5.4 Illustration of Υ^{-1} in image form, for a small size of image (32×32) for: (a) Uniform regularization, (b) Nonuniform regularization, $\lambda = 10$

The diagonal preconditioner for this approach will have the following form:

$$\mathbf{\Pi}_1 = \left(diag\left(\{\mathbf{Q}_0^T \mathbf{Q}_0 + \lambda \mathbf{\Delta}_{tot}\}^{-1}\right)\right)^{\frac{1}{2}} \tag{5.9}$$

5.3.2 Cholesky Decomposition

In designing the preconditioners $\mathbf{\Pi}$, so far, diagonal preconditioners have been used. These preconditioners are expected to improve the convergence by partly correcting the condition number. However, it is known that in all of the above preconditioners some approximations are made which their effect need to be verified and their efficiency need to be studied by numerical simulations and using real data.

Another alternative for designing $\mathbf{\Pi}$ is to use the exact inverse of the block-diagonal approximation to normal matrix with identical blocks $\mathbf{\Upsilon}$. The approximations involved in this approach is expected to be less than those made when designing the diagonal preconditioners in section 5.3.1.2 and 5.3.1.3 and therefore, this preconditioner is expected to perform better. However, the more precision in the approximation to normal matrix is obtained at the cost of requiring more memory for storing the preconditioning matrix and using a more complicated operation for applying this preconditioner to normal matrix. It should be noted that, as the blocks in $\mathbf{\Upsilon}$ are identical, only the elements of one block need to be stored and used for designing the preconditioner.

To apply this non-diagonal preconditioner to system, instead of calculating the exact inverse of $\mathbf{\Upsilon}$, one can solve a system of equations using direct methods. One of the methods which have been found efficient in solving systems of linear equations and is widely used for designing preconditioners, is the Cholesky decomposition. This method works by decomposing a Hermitian, positive-definite matrix, into the product of a lower triangular matrix $\mathbf{L}$ and its transpose $\mathbf{L}^T$ such that $\mathbf{L}\mathbf{L}^T = \mathbf{\Upsilon}$ (or equivalently product of a transpose of a upper triangular matrix $\mathbf{R}^T$ and itself $\mathbf{R}$ to get $\mathbf{\Upsilon} = \mathbf{R}^T\mathbf{R}$). The lower-triangular of Cholesky decomposition $\mathbf{L}$ for positive-definite matrices, is unique and for systems with symmetric matrix specifically, this method have been seen to outperform many other direct methods. Comparing it with other lower triangular-based methods, Cholesky decomposition have been found more efficient. For instance comparing it with LU-decomposition, this method when can be used, is almost twice as efficient as LU-decomposition for solving linear systems. Moreover, it has superior efficiency and numerical stability (Press *et al.*, 1992). This justifies this choice in this work[1].

In tomography problems, as discussed in section 5.1 the normal matrix is positive-definite and Hermitian. Moreover, it was seen that the dominant values are located on the diagonal blocks and so block-diagonal approximation to normal matrix, $\mathbf{\Upsilon}$, is also positive-definite and Hermitian. Therefore, one can use Cholesky decomposition to decompose $\mathbf{\Upsilon}$ to two triangular matrices.

The preconditioning matrix can then be chosen to be: $\mathbf{\Pi}_1 = \mathbf{L}^{-T}$, where $\mathbf{L}$ is unique for our problem and is the lower-triangular factor of first $K \times K$ block of block-diagonal approximation to normal matrix ($\mathbf{\Upsilon}$). Here, I have used the upper triangular factor of Cholesky decomposition for

[1] The procedure for calculating the lower triangular matrix $\mathbf{L}$ in Cholesky decomposition is discussed in (Bertsekas, 1999) and provided in Appendix E

designing preconditioner ($\mathbf{R} = \mathbf{L}^T$). The procedure will be similar to lower-triangular case. The preconditioner will then be $\mathbf{\Pi}_1 = \mathbf{R}^{-1}$.

Implementation: For applying this preconditioner to the problem and finding the minimum of the objective function iteratively, one needs to solve two systems of equations at every iteration. Note that considering the triangular structure of $\mathbf{R}$, these equations can be solved at a low cost. Comparing the operational cost of this preconditioner with the diagonal preconditioners, here one needs to solve two system of linear equations with size $K \times K$ at every iteration. The extra computation time per iteration needs to be numerically evaluated and compared.

5.3.2.1 Incomplete Cholesky factorization

Using the exact Cholesky decomposition for $\mathbf{\Upsilon}$ is possible when we are using the block diagonal approximation to normal matrix in 2D framework. This is because as discussed in section5.3, the diagonal blocks of this matrix are identical and one needs to use the elements of only one block for designing preconditioner. However, for 3D reconstruction, the size of this block will be multiplied by the number of slices and so storing this size of data may become costly or impossible. One alternative is to use a sparse approximation of $\mathbf{\Upsilon}$ for designing preconditioner. In making sparse approximation to this matrix, we are interested to find a good trade-off between the efficiency of the preconditioner in terms of improving the convergence and memory need. A preconditioner designed from a more sparse approximation to normal matrix will require less memory need and easier system of equations to be solved at every iteration at the cost of slower convergence.

Here, it was seen that the off-diagonal elements of $\mathbf{\Upsilon}$ are very close to zero. Therefore, it is expected that we do not lose a large accuracy by making sparse approximation to this block. The sparsity level which will be used for making sparse approximation is expected to have an inverse relationship with the time required per iteration. This relation will be explored with three different sparsity level experimentally in chapter 6.

Once the sparse approximation to normal matrix is obtained we can use incomplete Cholesky decomposition technique for designing preconditioners. For using incomplete Cholesky, the matrix which will be decomposed needs to be positive-definite and Hermitian. It is also known that the incomplete Cholesky decomposition can be found for diagonally dominant matrices. Here, the dominant elements are located on the diagonal and so there is no algebric problem with using this technique.

Implementation: The implementation of incomplete Cholesky is very similar to exact Cholesky but with an additional step: sparse approximation. For making sparse approximation, we will first preserve the diagonal elements as these are the dominant values. After calculating the maximum value of these elements, the elements with absolute value higher than a specific fraction of this value will be kept. The fraction which will be used here defines the sparsity level of preconditioner.

5.4 Approach 2: Block diagonalization of normal matrix using FFT

In section 5.1 it was seen that the normal matrix is BC. Moreover, in section 5.2 it was discussed that using FFT method, one can find the block-diagonalization of a BC matrix. The inverse of this specially structured matrices can then be calculated in a fast way and so used for designing preconditioners.

In Appendix D, definition of circulant matrices is given and a detailed review on how the FFT method can be used to diagonalize the circulant matrices and used as preconditioner is provided. The same approach can be followed for block-diagonalization of a BC matrix.

Assuming that the BC normal matrix is represented by $\mathbf{C} = \mathbf{P}^T\mathbf{P} + \lambda\mathbf{D}^T\mathbf{D}$, the relationship between this matrix and it block-diagonalized form ${}^{bd}\mathbf{C}$ can be represented using following expression:

$$\mathbf{C} = \mathbf{F}^{-1\,bd}\mathbf{C}\mathbf{F} \tag{5.10}$$

where $\mathbf{F}$ and $\mathbf{F}^{-1}$ represent the Fourier and inverse Fourier transform matrices[1] respectively. It should be noted that according to the Bochner's theorem, Fourier transform preserves the positive-definiteness of the problem meaning that Fourier transform of a positive-definite function is positive-definite. The normal matrix here is Hermitian and positive-definite, so these properties are preserved when using FFT for diagonalization.

Once, the normal matrix is block diagonalized using FFT, the resulting matrix (${}^{bd}\mathbf{C}$) has N blocks of size $K \times K$. To check the properties of this new matrix, the L_2 norm of each block was calculated. Fig.(5.5) shows these norms versus block numbers for a 512×512 image when the object is discretized in polar coordinates. We can see from this figure that, unlike what was observed in the spatial domain, these blocks are not identical and the highest norm corresponds to the block in the middle $\theta_n = \frac{N}{2}$. Consequently, we cannot treat all the blocks in the same manner.

Using (5.10), $\mathbf{C}^{-1}$ can be easily calculated as following and used for designing preconditioner:

$$\mathbf{C}^{-1} = \mathbf{F}^{-1\,bd}\mathbf{C}^{-1}\mathbf{F} \tag{5.11}$$

It should be noted that in using (5.11) for BC matrix $\mathbf{C}$, we benefit from the fact that the inverse of a block diagonal matrix is also block diagonal, with each block being equal to the inverse of the corresponding block in the matrix.

The resulting matrix ${}^{bd}\mathbf{C}^{-1}$ is the exact inverse of the normal matrix and using it as preconditioner, the iterative optimization algorithms converge in one iteration only. However, it should be noted that having to store the block-diagonal Hessian matrix may be expensive even for 2D recon-

[1] See chapter 1 of (Chen, 2005) or any other reference for the definition of Fourier transform matrix. Note that $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$ where N is the size of problem and $\mathbf{F}^*$ represent the conjugate transpose of Fourier matrix.

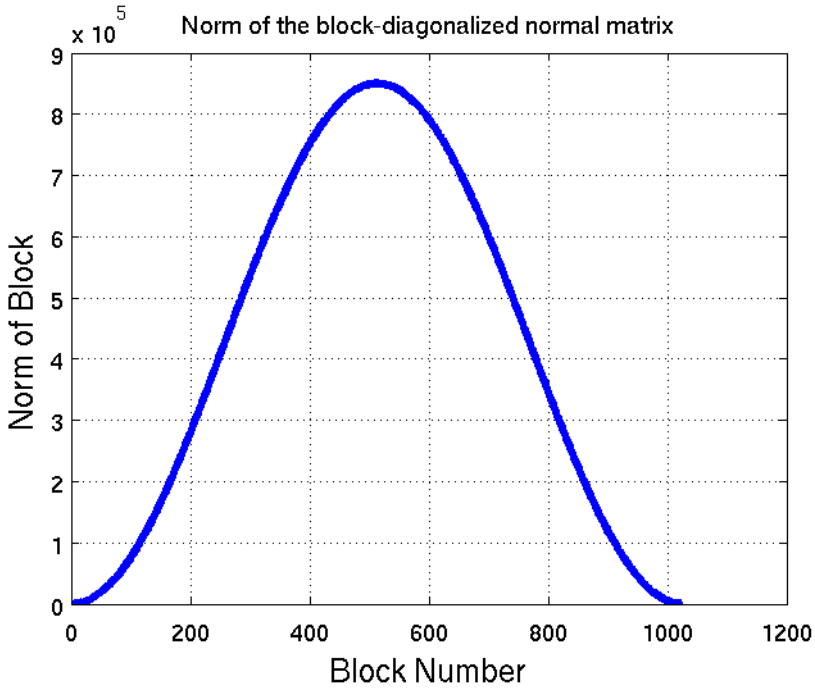


Figure 5.5 Norm of blocks of the normal matrix.

struction with realistic data size. Consequently, some approximation to this matrix is required to make this method practically usable.

In general, when using the BC matrices as preconditioner we are interested in computing the product of this matrix or of its inverse with a vector quickly. Using FFT, the forward and backward operations ($\mathbf{f} = \mathbf{C}\mathbf{g}$ and $\mathbf{g} = \mathbf{C}^{-1}\mathbf{f}$) can be performed quickly for any vector ($\mathbf{f}$ or $\mathbf{g}$). These operations are used at every iteration when implementing the solver with preconditioner.

To be more precise, for computing the above quantities for BC matrix $\mathbf{C}$, we can use what was obtained for circulant matrices in (D.11) and (D.12) and perform these operations as follows:

$$\mathbf{f} = \mathbf{C}\mathbf{g} = \mathbf{F}^{-1\,bd}\mathbf{C}\mathbf{F}\mathbf{g} = \mathbf{F}^{-1}[{}^{bd}\mathbf{C} \times \mathbf{F}\mathbf{g}] \tag{5.12a}$$

$$\mathbf{g} = \mathbf{C}^{-1}\mathbf{f} = \mathbf{F}^{-1\,bd}\mathbf{C}^{-1}\mathbf{F}\mathbf{f} = \mathbf{F}^{-1}[{}^{bd}\mathbf{C}^{-1} \times (\mathbf{F}\mathbf{f})] \tag{5.12b}$$

It should be noted that in the above equations, operations are performed from right to left. The Fourier transform of vectors $\mathbf{f}$ and $\mathbf{g}$ can easily be obtained using FFT technique with negligible computational cost[1]. For efficient implementation of (5.12b), multiplication of ${}^{bd}\mathbf{C}^{-1}$ by any vector needs to be performed in a systematic way to avoid high calculation cost. This will be done by

[1]Note that using FFT method for calculating the Fourier transform of a vector requires at most $O(N\ log\ N)$ operations as oppose to $O(N^2)$ where N represents the size of the vector and O denotes an upper bound. The gain becomes more evident when N is large. This property is a well-known property of FFT method.

factorizing ${}^{bd}\mathbf{C}$ to some diagonal or triangular matrices and then solving two systems of equations at each iteration (solve ${}^{bd}\mathbf{C}\mathbf{h} = \mathbf{F}\mathbf{f}$ with ${}^{bd}\mathbf{C}$ being diagonal or triangular, instead of $\mathbf{h} = {}^{bd}\mathbf{C}^{-1} \times (\mathbf{F}\mathbf{f})$). In this way, the multiplications can be performed in a fast way with low numerical lost and we do not need to calculate the ${}^{bd}\mathbf{C}^{-1}$ explicitly.

As explained in section 5.2, the exact block-diagonal factorization of $\mathbf{C}$ can not be used for preconditioning purposes in practice due to the large size of the real data. Therefore, a diagonal or sparse approximation to the ${}^{bd}\mathbf{C}$ needs to be computed without having to evaluate the complete ${}^{bd}\mathbf{C}$.

In calculating ${}^{bd}\mathbf{C}$ we are using Fourier and inverse Fourier transform as expressed in (5.10). These operations are applied to the matrix $\mathbf{C}$ in a vector-wise manner. This mean that for instance the first elements of all the diagonal blocks of ${}^{bd}\mathbf{C}$ are calculated by using FFT operation on the first block-row of $\mathbf{C}$. Therefore using these operations ${}^{bd}\mathbf{C}$ can be computed element-wise or diagonal-wise.

The possibility of calculating ${}^{bd}\mathbf{C}$ in a diagonal-wise form using FFT method is an essential property which makes this method practically useable. This is because given the large size of the real data, making sparse approximation to this matrix is not practical if one needs to compute the complete block-diagonal normal matrix ${}^{bd}\mathbf{C}$.

The procedure of computing the FFT-based preconditioners and how they are used with numerical methods is demonstrated as flowchart in Fig.(5.6).

5.4.1 Diagonal Preconditioner

One simple approximation to the block-diagonalized normal matrix ${}^{bd}\mathbf{C}$, is to consider its diagonal elements only. The diagonal preconditioning matrix $\mathbf{\Pi}$ will then be the inverse of this matrix, $\mathbf{\Pi} = \mathbf{F}^{-1}\{diag({}^{bd}\mathbf{C})\}^{-1}\mathbf{F}$ (note that the inverse of this matrix is simply the inverse of each of its diagonal elements). As the diagonal enteries of the normal matrix are positive, this assures that the preconditioning matrix will always be non-singular.

For the implementation aspect, this preconditioner is affordable. This is because only one needs to store a vector of size $KN \times 1$ and the cost associated with applying it in every iteration is negligible as it consists of a FFT operation, two simple element-by-element vector products and an inverse FFT per iteration.

It should be emphasized again that by using this approximation in designing preconditioner, we are eliminating all the elements which are located off-diagonal. This will affect the convergence and the number of iterations required for reconstruction. The impact of this approximation needs to be experimentally evaluated.

Implementation: For implementation of diagonal preconditioner using this approach, the main diagonal of ${}^{bd}\mathbf{C}$ will be calculated as discusses in section 5.4. The inverse of this diagonal matrix

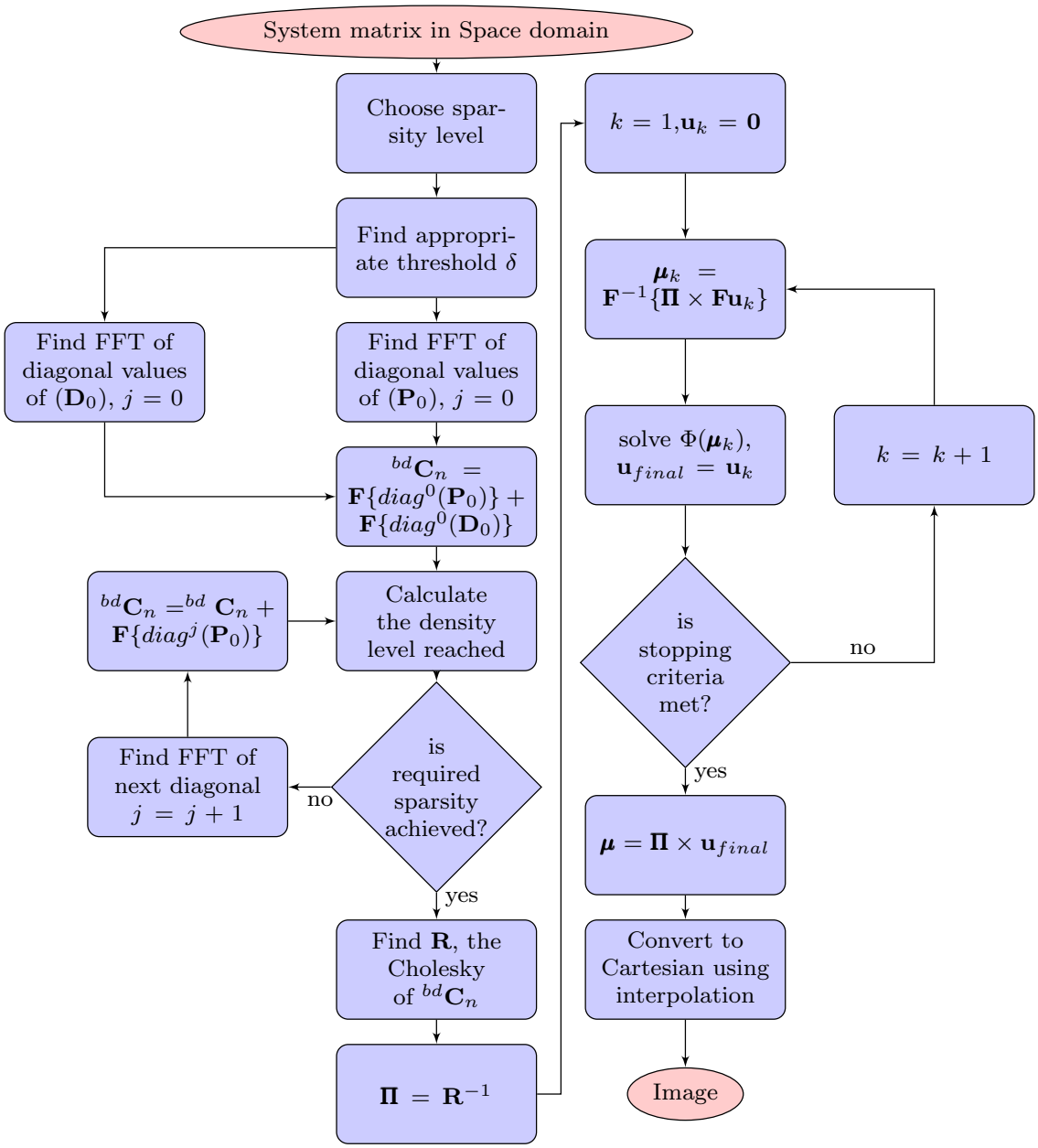


Figure 5.6 Flowchart for implementation of preconditioner based on FFT method and using it with iterative method. Note that here ${}^{bd}\mathbf{C}$ represents the block diagonalized normal matrix, $diag^k(\mathbf{P}_0)$ and $diag^k(\mathbf{D}_0)$ is the k-th diagonals of matrix $\mathbf{P}_0$ and $\mathbf{D}_0$ with $k = 0$ referring to main diagonal, where $\mathbf{D}_0$ denote the first K rows of sum of matrices of differences. $\mathbf{R}$ is the upper triangular matrix of Cholesky method, μ is the attenuation map of the reconstructed object and Φ represents the objective function.

will be used for preconditioning. For applying this matrix to the Hessian of the objective function and multiplying it with any vector, expressions in (5.12) will be used.

5.4.2 Cholesky Decomposition

Another approach for designing preconditioners is to use the Cholesky decomposition as explained in section 5.3.2. This method works by finding an upper-triangular matrix ($\mathbf{R}_i$) for each block of ${}^{bd}\mathbf{C}$ such that: $\mathbf{R}_i^* \mathbf{R}_i = {}^{bd}\mathbf{C}^i$, where ${}^{bd}\mathbf{C}^i$ denotes the i-th block of ${}^{bd}\mathbf{C}$, $1 \leq i \leq N$, and has size $K \times K$ and * shows the conjugate transpose operation. ${}^{bd}\mathbf{C}$ can then be shown as below:

$$ {}^{bd}\mathbf{C} = \begin{bmatrix} {}^{bd}\mathbf{C}^1 & & & \mathbf{0} \\ & {}^{bd}\mathbf{C}^2 & & \\ & & \ddots & \\ \mathbf{0} & & & {}^{bd}\mathbf{C}^N \end{bmatrix} \tag{5.13} $$

It is known that the lower triangular Cholesky matrices are usually full. This is usually the case even if the matrix which will be decomposed is sparse. Consequently, for preconditioning, this would require to store N matrices which will result in storing about $N \times K^2/2$ elements which can become impossible for practical cases. Consequently, use of exact Cholesky is avoided in practice.

The solution to this problem, similar to what has been already mentioned in section 5.3.2.1, is to use the incomplete Cholesky factorization. To do this, we need to make a sparse approximation to ${}^{bd}\mathbf{C}$.

By making a sparse approximation to ${}^{bd}\mathbf{C}$, and denoting it by ${}^{bd}\widetilde{\mathbf{C}}$, we can use the incomplete Cholesky method for each block ${}^{bd}\widetilde{\mathbf{C}}_n^i$, and consequently $\mathbf{R}_i$, the upper triangular matrix, will have the same sparsity level as the corresponding ${}^{bd}\widetilde{\mathbf{C}}^i$. This will reduce the memory required for storing the $\mathbf{R}_i$s. The preconditioning matrix will then be block diagonal as below:

$$ \mathbf{\Pi} = \begin{bmatrix} \mathbf{\Pi}_1 & & & \\ & \mathbf{\Pi}_2 & & \\ & & \ddots & \\ & & & \mathbf{\Pi}_N \end{bmatrix} = \mathbf{F}^{-1} \begin{bmatrix} \mathbf{R}_1 & & & \\ & \mathbf{R}_2 & & \\ & & \ddots & \\ & & & \mathbf{R}_N \end{bmatrix}^{-1} \mathbf{F} \tag{5.14} $$

Using sparse approximation to ${}^{bd}\mathbf{C}^i$ will result in having simpler linear equations to solve using the direct method at every iteration and therefore, this may decrease the time required per iteration. The reduction in time per iteration may compensate for the approximations made by using the incomplete Cholesky decomposition as oppose to exact one. This should be experimentally verified and the effect of these approximations should be studied in terms of total time required for reconstruction and number of iteration needed for convergence.

To summarize, the sparse approximation is expected to be a trade-off between the memory need and reconstruction time. This will be further discussed when results are compared in chapter 6.

Implementation: Using the discussion in section 5.4, a sparse approximation to ${}^{bd}\mathbf{C}$ is first obtained. This gives a sparse block-diagonal matrix. Then the Cholesky decomposition of this matrix is calculated and used as preconditioner using (5.14). Similar to diagonal preconditioner explained in previous section, multiplication of this matrix by any vector is done in Fourier domain and using equations (5.12).

5.4.2.1 Sparse approximation to ${}^{bd}\mathbf{C}$

Here, two approaches are followed for making the sparse approximation to the block diagonalized normal matrix. The first approach is to generate the main diagonal elements of ${}^{bd}\mathbf{C}$ first, and compute their maximum modulus. A fraction of the maximum moduluses of the diagonal elements will then be used as threshold and the elements with absolute value higher than this value will be kept. This fraction will be chosen such that the desired sparsity level is achieved. This approach will be referred to as "Global sparse approximation".

The second approach is to make the sparse approximation in a block-wise manner. In this approach a threshold specific to each block is selected as a constant fraction of the magnitude of the largest element of each block and the elements of each block are preserved based on the threshold of the corresponding block. This method is called "Block-wise sparse approximation" here.

The performance of preconditioners obtained by these two approaches will be compared in chapter 6 to choose the most efficient one. In making the comparisons, one should make sure that thresholds are chosen such that both methods provide matrices with the same sparsity levels. This makes the comparisons fair for drawing a correct conclusion on which method to choose for making the sparse approximation.

5.5 Conclusion

In this work two approaches are followed for designing preconditioners. The first approach consists of making a block-diagonal approximation to the normal matrix and then using either diagonal preconditioners or Cholesky preconditioners. This approach is expected to be highly affected by approximations and therefore, the efficiency of such preconditioners needs to be studied carefully.

The alternative approach is to make use of the special structure of the normal matrix, i.e., BC, and find a block diagonalized version of it. We can then use either diagonal preconditioners or Cholesky methods for improving the convergence. In using this method it should be noted that the blocks will not be identical unlike the first approach and consequently all the blocks need to be considered. This means that the memory need for this approach will be N-times higher. Therefore, for an image with K number of radial samples and N number of angular samples, the number of

elements which need to be stored and used as preconditioner for each of these approaches follow the table below:

Table 5.1 Number of nnz elements stored for each preconditioner.

Method used	Preconditioner	Number of non-zero elements stored
Approach 1	Diagonal	K
	Cholesky	$1/2 \times K \times (K+1)$
Approach 2	Diagonal	$K \times N$
	Cholesky	$1/2 \times K \times (K+1) \times N$

In the Cholesky row in above table, it is assumed that the exact Cholesky decomposition is used. In practice incomplete Cholesky decompositions are used and by choosing an appropriate sparsity level. This will reduce these numbers accordingly.

We expect the preconditioners of second approach to perform better as there are less approximations involved for this case. We also expect the Cholesky preconditioner to be more efficient compared to diagonal preconditioner of same approach. This is due to the lower level of approximations involved in the former. These points will be experimentally verified in the next chapter.

CHAPTER 6

RESULTS AND DISCUSSION

The goal of this chapter is to validate the methodology developed in chapters 4 and 5, and the efficacy of the proposed methods in addressing the problems associated with using a polar grid for discretization of the object. These problems, which have been summarized in chapter 3, include: what is the best scheme for discretization of the object in polar coordinates in terms of implementation, how does polar discretization of the object affects the conditioning of the normal matrix and so the convergence of the iterative algorithms, and how to express the penalty function in polar coordinates.

The eigenvalue spectrum of the normal matrix for standard discretization (Cartesian) and the polar discretization in Fig.(4.2) show that using polar coordinates, the eigenvalue spectrum is deteriorated and the eigenvalues are spread over a range significantly wider than that of the Cartesian representation.

In the last two chapters, we attempted to address some of these problems : A simple tessellation scheme was introduced in chapter 4 for discretization of the object in polar coordinates and two approaches for penalization of neighboring pixels were discussed. The eigenvalue spectrum was seen to be more deteriorated when nonuniform weighitng, which is consistent with the definition of the gradient, is used as penalty function (Fig.(4.3)). This emphasizes the necessity for developing a technique which corrects the conditioning for this type of discretization.

For this purpose different types of preconditioners were proposed, in chapter 5. This was done by taking advantage of the special structure of the normal matrix. The efficiency of these preconditioners will be studied and verified experimentally in this chapter using some realistic assumptions for the parameters.

6.1 Parameters and model used

For consistency and to be able to make fair comparisons, all the experiments performed in this chapter were conducted on realistic-size simulated data: $(512, 512)$ images, 512 equally spaced detectors, 1024 projections per rotation. The size of the polar grid used for discretization of the object (μ) is $(256, 1024)$ and covers a disk whose diameter is equal to the side-length of the square Cartesian image. The numbers used here are close to what has been used in real scanners (number of projections 1160 and number of detectors 672). These numbers are chosen to be powers of 2 to allow easily expansion to other sizes of data for comparison purposes.

To avoid an "inverse crime" situation (Mueller et Siltanen, 2012), that may bias the results, the sinogram was generated using a forward model different from the one used to perform the reconstructions. This forward model is described in section 2.5.2 and operates on the object discretized in Cartesian coordinates, which is rotated and interpolated at each projection angle.

Here, it is assumed that the additive noise can be modeled as a Gaussian distribution with constant variance ($\mathbf{\Sigma}_b = \mathbf{I}$). Except for section 6.2 where the behavior of the objective function for different SNR levels is studied, in all the experiments of this chapter, Gaussian noise was added to the sinogram (logarithm of the photon counts) so as to obtain a global 30 dB signal-to-noise ratio.

Sparse projection operators $\mathbf{P}_0$ and $\mathbf{Q}_0$ were computed and stored using an incremental CRS scheme (Koster, 2002). Encoding of the sparse matrices and sparse matrix-dense vector products was done in C^{++} using Albert-Jan N. Yzelman's publicly available SparseLibrary [1]. Parallelization was performed in an elementary manner using the OpenMP® library and the algorithm developed in section 4.4 (Goussard *et al.*, 2013).

Regularization function $R(\mu)$ consists of a L_2 penalty applied to all first differences of μ over first-order neighborhoods, with uniform and non-uniform weighting. Maximization of the resulting criteria was performed using the L-BFGS algorithm from the quasi-Newton family. The justifications for these choices are given in chapter 3.

Experiments were performed on an Intel multi-threaded 12-core processor running Linux and the Shepp-Logan phantom was used as numerical phantom. This is a standard test phantom which is widely used in the litreature for evaluating the developed algorithms.

The figure of merit which was used in this work for evaluating the performance of the proposed methods is the normalized distance to the exact solution. As already stated, using a quadratic regularization function, this problem has a closed-form solution. Using the FFT technique, the exact solution can be calculated using following expression:

$$\hat{\boldsymbol{\mu}} = \mathbf{F}^{-1}\{\mathbf{P}^T\mathbf{\Sigma}\mathbf{P} + \lambda\mathbf{D}^T\mathbf{D}\}\mathbf{F}\mathbf{y} \tag{6.1}$$

where $\mathbf{F}$ and $\mathbf{F}^{-1}$ represent Fourier and inverse Fourier transform operations (discussed in section 5.4).

Note that, in a 2D framework, computation of the exact solution is possible on current, PC-type computers. This becomes impossible in 3D reconstruction due to the larger size of data. Convergence speed is used to assess the behavior of optimization problem in this work. To do this, the exact solution is used in polar coordinates ($\hat{\boldsymbol{\mu}}(r,\theta)$). This is becasue, the value of $\boldsymbol{\mu}$ in every iteration is updated in polar coordinates and therefore calculating the normalized distance in same coordinates avoid introducing interpolation errors in assessing quality of the results.

[1] Library and documentation available at http://people.cs.kuleuven.be/ albert-jan.yzelman/software.php.

6.2 Noise level

We first study the behavior of this problem to different SNR levels to see if the pattern remains the same. If it is the case, then we can fix the noise level and perform all the experiment with same assumption and expect a similar behavior with different SNRs.

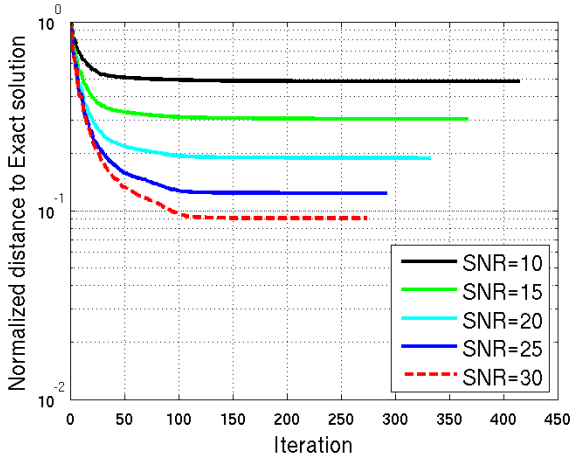


Figure 6.1 Convergence for uniform regularization with two different SNRs.

Fig.(6.1) shows the convergence plots for five different SNR levels (10-30 dB), when uniform weighting is used for regularization. Looking at this figure we can see that the pattern remains the same for all SNR values.

A similar behavior is seen in Fig.(6.2) when nonuniform weighting is used for regularization. These results suggest that we can fix the SNR used for the experiments and expect a similar behav-

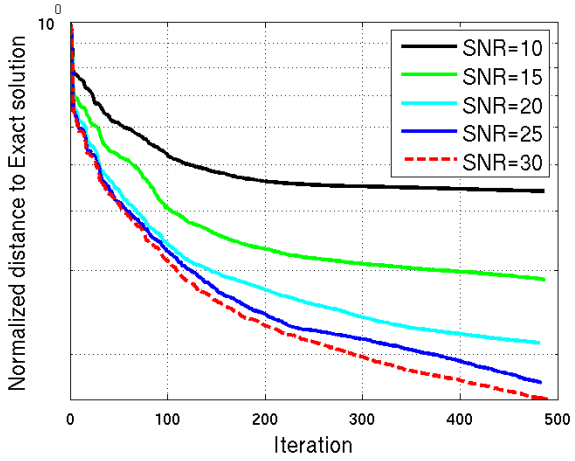
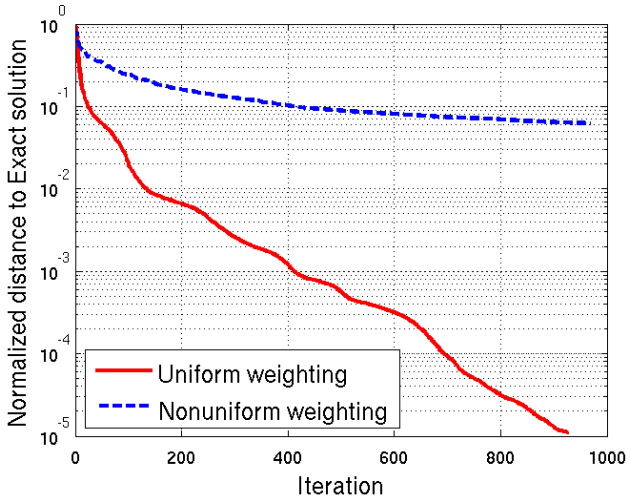


Figure 6.2 Convergence for nonuniform regularization with two different SNRs.

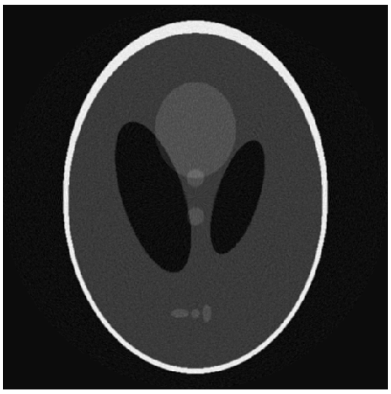
ior. In the rest of this chapter the SNR will be set to 30dB.

6.3 Regularization Function

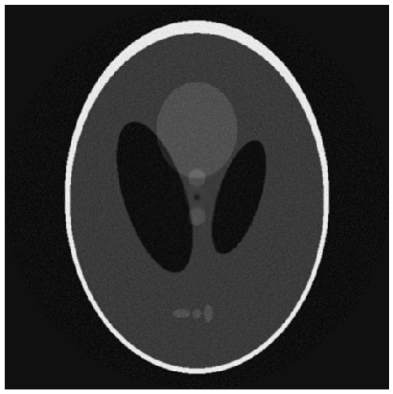
As discussed in section 4.5, it is not clear how the neighboring pixels need to be penalized when the object is discretized in polar-coordinates. In this section, this question will be answered experimentally through simulations. We will first study the effect of each type of penalty functions on convergence and explore the quality of reconstructed images in each case. We will then proceed to the next section where the performance of different type of preconditioners are studied.



(a)



(b)



(c)

Figure 6.3 Regularization function performance: (a) L_2 distance to the exact solution, (b) uniform weighting, (c) nonuniform weighting.

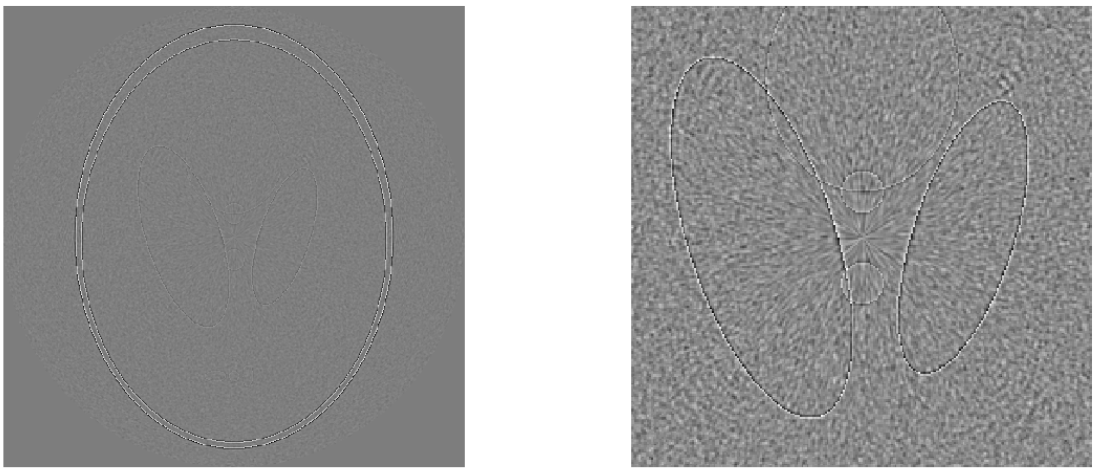
Here, the maximum number of iterations for the L-BFGS algorithm is set to a high value (i.e.,

1000) to study the convergence behavior of the two regularization functions. All other stopping criteria which can be used for L-BFGS algorithm to terminate, are chosen such that the termination is controlled by the maximum number of iterations, only.

Fig.(6.3a) shows the normalized distance to the exact solution when each type of regularization functions have been used. The corresponding reconstructed images are presented in Fig.(6.3b) and Fig.(6.3c). From Fig.(6.3a), one can see that when uniform weighting is used, the algorithm converges significantly faster than the nonuniform case. The algorithm with nonuniform regularization has not converged after 1000 iterations (Fig.(6.3c)). In using nonuniform weighting, pixels are weighted by the inverse of their distance to the neighboring pixels and so a high weight is applied to those which are located closer to the center of image. This makes the convergence slower for these pixels. This can be seen in (Fig.(6.3c)).

Using this experiment and the eigenvalue spectrum for each type of regularization functions(Fig.4.3) we can conclude that using nonuniform regularization for polar discretization significantly deteriorate the condition number and the algorithm does not converge in absence of preconditioners. However, this type of regularization is the one which is consistent with the physical meaning of the penalty term.

Now, to evaluate the performance of the reconstructed image using uniform weighting, the difference of the reconstructed image using this penalty function and the actual numerical phantom is plotted in Fig.(6.4). Taking a close look at the center of the difference of the reconstructed



(a) (b)

Figure 6.4 Difference with actual phantom for uniform regularization: (a) complete image, (b) smaller graphical window.

image and the actual phantom (Fig6.4a) in Fig.(6.4b) we can see that there are some streak artifacts present in the reconstructed image. This may be due to the approximations which are made by using uniform weights. These streaks become more evident when SNR is high. In the other hand, it was seen that this type of penalty function has a shorter reconstruction time, therefore, one can try to further decrease the reconstruction time for this penalty term by designing appropriate preconditioners.

6.4 Preconditioning

In this section performance of the different preconditioners designed in chapter 5 are evaluated. To do this, we have first assessed the effect of additional approximations which have been made by using incomplete Cholesky decomposition versus exact Cholesky decomposition in section 6.4.1 for the first approach. We will then discuss the parameters which are involved in designing preconditioner using second approach (block-diaonalization of the normal matrix) in section 6.4.2.

6.4.1 Approach1: Block-diagonal approximation to the normal matrix

In chapter 5, three different preconditioners were designed by making block-diagonal approximation to the normal matrix. Moreover, it was seen than one can use a non-diagonal preconditioner using exact or incomplete Cholesky decomposition.

In this section we are interested in comparing exact Cholesky and incomplete Cholesky preconditioners, to study the gain achieved by making the sparse approximation when using incomplete Cholesky decomposition and the associated reduction in the convergence speed. This study will be done for the block-diagonal approximation to the normal matrix where exact Cholesky decomposition is also affordable for 2D problems. We will study the effect of approximations on convergence when using the incomplete Cholesky decomposition.

As explained in section 5.3.2, the Υ blocks are almost full. So, if one wants to use incomplete Cholesky decomposition, one will first need to make a sparse approximation to these blocks. Fig.(6.5) shows the relationship between sparsity (density level) and the threshold value used for keeping the largest elements when nonuniform weighting is used. One can see that for keeping 30% or less elements, the threshold (δ) needs to be chosen to be in the range $10^{-7} \leq \delta \leq 10^{-5}$. For $\delta > 10^{-5}$ sparsity remains the same. This is because as already stated off-diagonal elements of $\boldsymbol{\Upsilon}$ are very close to zero and their magnitude is significantly smaller than that of the diagonal enteries.

Here, the incomplete Cholesky preconditioners with three different sparsity levels equal to 0.3, 0.05 and 0.004 (thresholds are set to be 10^{-7}, 10^{-6} and 10^{-5} respectively) have been used. These values allow to study the effect of sparse approximations for a very loose case (0.3) to a strike case (0.004). It is expected that number of iterations required for convergence increases with the sparsity

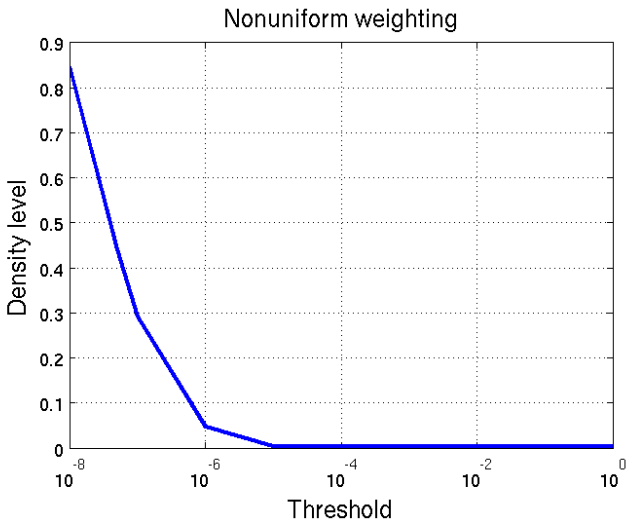
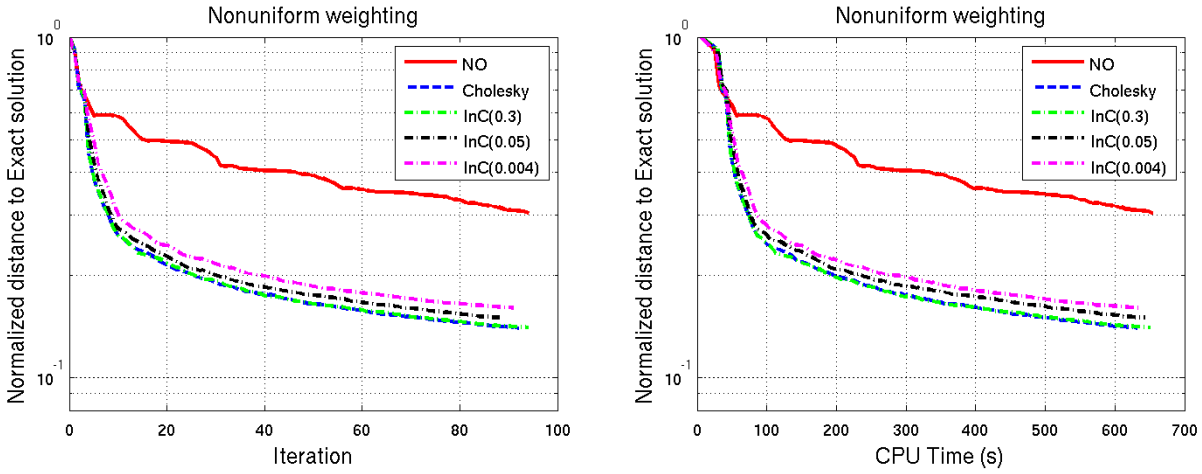


Figure 6.5 Sparsity versus threshold.

level. The results are presented in Fig.(6.6). Here, the stopping criteria is the maximum number of iterations which has been chosen to be 100. Moreover, the maximum number of calculations of objective function and the gradient is set to the same number as the maximum number of iterations. This allows to compare the efficiency of the preconditioners for same reconstruction time.



(a) (b)

Figure 6.6 Comparison between the performance of preconditioners from exact Cholesky decomposition on block diagonal approximation to normal matrix and the incomplete Cholesky with 3 different sparsity level. Normalized distance to exact solution :(a) Convergence(iteration number), (b) Convergence (reconstruction time)

In Fig.(6.6a), the normalized distance to the exact solution is plotted versus iteration numbers for these preconditioners as well as the no preconditioning case. Both type of preconditioners have

improved the convergence compared to when no preconditioner have been used. Comparing the performance of exact Cholesky versus incomplete Cholesky-based preconditioners, one can see that here exact Cholesky preconditioner and the incomplete Cholesky preconditioner behave very similarly for sparsity levels of 30% and less. For the lowest density level (0.004), the incomplete Cholesky preconditioner shows a slower convergence however the difference is not significant. Another point which needs to be underlined here is that the number of iterations for the optimization problem to converge is almost the same for all the preconditioners.

Now, to see the effect of sparsity on the reduction of time required per iteration, the normalized distance to the exact solution for each preconditioner is plotted versus the CPU time in Fig.(6.6b). One can observe that the reduction of time per iteration in sparse incomplete Cholesky preconditioners is negligible and overall, the time required for the algorithm to stop is almost the same for all the preconditioners. Moreover, comparing the normalized distance to the exact solution for each preconditioner when it has been stopped, we can see that the exact Cholesky decomposition has the smallest distance followed by the least sparse incomplete Cholesky preconditioner.

Consequently, we can conclude that if the sparsity level is chosen wisely, the reduction in the convergence speed of the incomplete Cholesky preconditioner will be negligible and one can gain up to 70% in the number of elements needed to be stored. One can then choose the preconditioner with the smallest memory footprint which perform very similarly both time-wise and distance-wise.

Similar results were observed for uniform weighting regularization.

6.4.2 Approach 2: Block diagonalization of the normal matrix using FFT

In chapter 5, it was seen that considering the special structure of the normal matrix, one can block-diagonalize this matrix using the FFT method. The inverse of this block-diagonalized matrix can then be used for preconditioning purposes. The efficiency of the preconditioners designed using this approach is to be verified experimentally in this section.

As it was discussed, storing the complete block-diagonal normal matrix (${}^{bd}C$) can become non-affordable for practical cases such as 3D imaging and therefore some approximations to this matrix are required. In this section, we will first discuss the sparsity level which will be used for designing preconditioners and the sparse approximation method which perform better. We will then study the effect of this sparse approximation on the properties of normal matrix. The performance of preconditioners designed using this approach will then be compared.

6.4.2.1 Sparsity level

The sparsity level is chosen so as to obtain a good trade-off between the memory need for storing these elements and the reconstruction time. Assume that the density level for the preconditioning

matrix $\mathbf{\Pi}$ is defined as the ratio of the number of nonzero elements to the maximum number of nonzero elements in the block diagonal normal matrix ($K \times K \times N$). We are interested in comparing the performance of two preconditioners based on incomplete Cholesky decomposition with density levels equal to twice and four times of that of the diagonal preconditioner proposed in section 5.4.1. Choosing these density levels allows comparing the effect of having a more precise preconditioner where the required memory footprint is still affordable.

For a realistic data size with 512×512 pixels, a diagonal preconditioner will be 2.09 MB. Sparse preconditioners with density level equal to twice and four times of that of diagonal will then require 4.1MB and 8.4MB of memory, respectively. Increasing the number of non-zero elements in preconditioner will not only increase the memory need, it will also affect the reconstruction time per iteration and increase the time required for generating the approximated block-diagonal normal matrix. Therefore, a preconditioner which can provide a good convergence speed with the least memory footprint is sufficient.

6.4.2.2 Sparse approximation to ${}^{bd}\mathbf{C}_n$: global or block-wise?

In section 5.4, the procedure for calculating the block-diagonalization of normal matrix was explained and the necessity of making sparse approximation to this matrix was discussed. Two methods were described in section 5.4.2.1 for making sparse approximation to normal matrix, global and block-wise.

In this section, the goal is to choose which of these methods is more efficient when used as preconditioner. This needs to be verified experimentally for making solid conclusion as there are approximations involved. This experiment was done for both type of regularizations: nonuniform and uniform weighting. Results will be discussed below.

Note that, according to Remark 5.4.2.1, in the computation of the incomplete Cholesky factors, the diagonal values of each block may be increased by a factor of α in order to preserve positive-definiteness. In the following, the value of α is chosen to be the minimum possible value which makes the matrix of interest positive-definite.

For fair comparison of the performance of global and block-wise methods, thresholds need to be chosen such that same sparsity level is achieved.

For both regularization function, two sparse preconditioners are used with sparsity level equal to half and one forth of diagonal preconditioner. This corresponding to one half and one fourth of diagonal preconditioners are respectively equal to 0.0078 and 0.0156.

Nonuniform weighting

Table(6.1) summarizes the parameters used in this experiment when nonuniform weighting is used. The performance of these preconditioners is compared in Fig.(6.7). The preconditioner

with lower sparsity level (higher density) has a better performance. This is because there are less approximations made in designing the preconditioner at the cost of higher memory need (doubling the number of stored elements).

Table 6.1 Nonuniform regularization: parameter values.

		Method	
		Global	Block-wise
Density=0.0078	δ	4.5×10^{-6}	2×10^{-2}
	α	6×10^{-1}	3×10^{-1}
Density=0.0156	δ	1.8×10^{-6}	5×10^{-4}
	α	1×10^{-3}	1.2×10^{-1}

One can see from these figures that the block-wise sparse approximation is performing better than the global method for both sparsity levels. Therefore, we can conclude that preconditioners

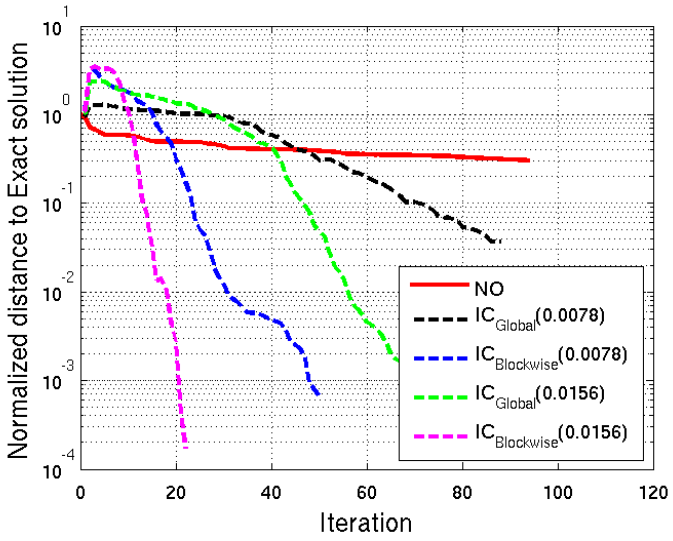


Figure 6.7 Nonuniform weighting, comparison of the performance of the global and block-wise sparse approximations.

with the block-wise sparse approximation are more effective than the global method, when nonuniform weighting is used for regularization.

Uniform weighting

In order to check the consistency of the results, the above experiments were repeated in the case of uniform weighting. Table(6.2) summarizes the parameters used for this experiment.

Fig.(6.8) illustrates the performance of these preconditioners. For the first sparsity level (density= 0.0078), it is clear that the block-wise method is performing better. However, for the second sparsity level, neither approach shows any clear advantage.

Table 6.2 Uniform regularization: parameter values.

		Method	
		Global	Block-wise
Density=0.0078	δ	6.5×10^{-6}	1.5×10^{-1}
	α	6×10^{-1}	1.5
Density=0.0156	δ	3×10^{-2}	8×10^{-2}
	α	0	8×10^{-1}

The reason may be that for global method the sparse matrices are positive-definite and diagonally dominant, so there is no need to introduce additional approximations by using α, as opposed to the block-wise case.

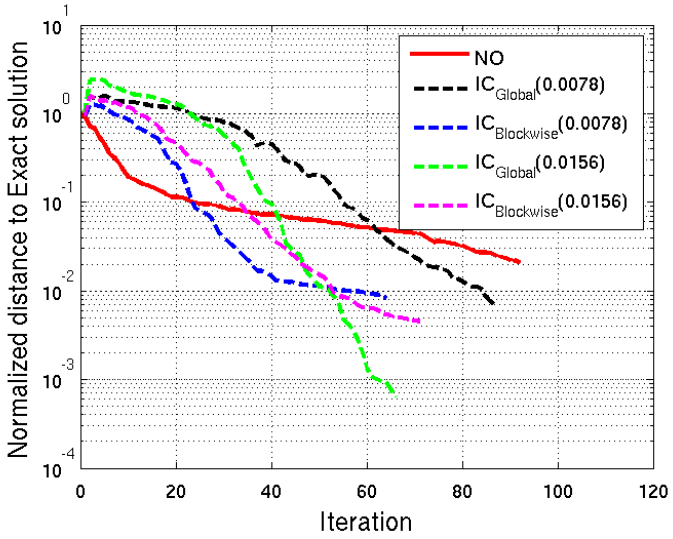
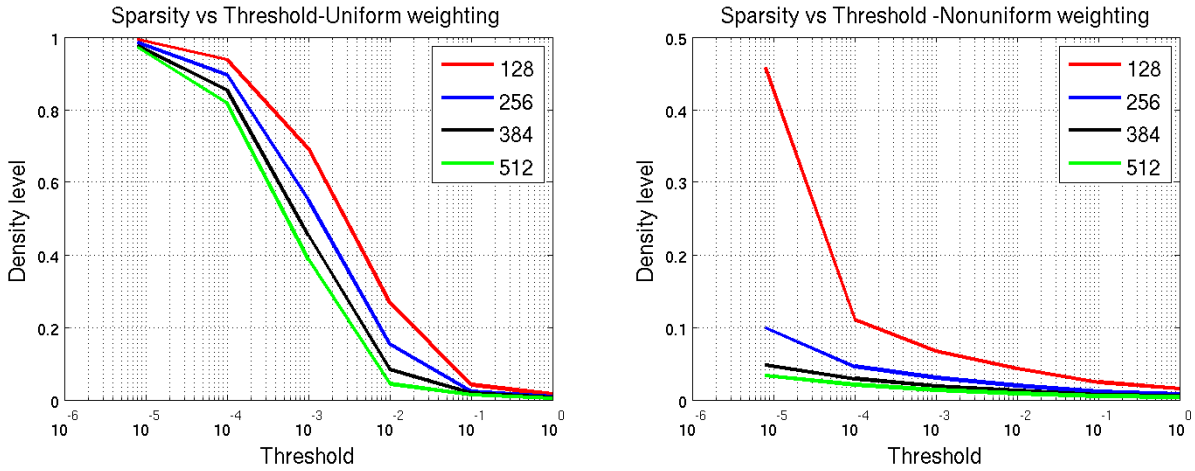


Figure 6.8 Uniform weighting, Comparison of the performance of the global and block-wise sparse approximation as preconditioner.

To summarize, the experiments with both regularization geometry suggests that in most of the cases, the block-wise approximation performs better and therefore this method will be used for designing the preconditioners.

6.4.2.3 Sparsity versus threshold

In section 6.4.2.2 it was seen that in most of the cases, preconditioning using incomplete Cholesky performs better when using block-wise method for making sparse approximation to block-diagonalized normal matrix. Fig.(6.9) shows the relationship between the threshold and the sparsity level for both



(a) (b)

Figure 6.9 Sparsity level for different thresholds: (a) Uniform, and (b) Nonuniform weighting

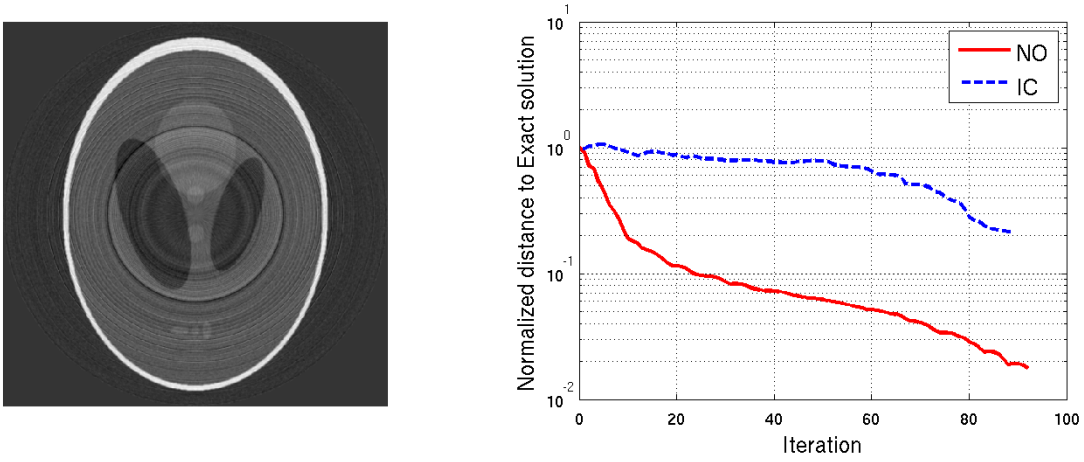
uniform and nonuniform regularization and for four different image sizes. These charts can be used to choose the values of δ for the desired sparsity levels.

6.4.2.4 Incomplete Cholesky decomposition and the effect of α

Sparse approximation to the block-diagonalized normal matrix (${}^{bd}\mathbf{C}$) can destroy its positive-definite property for some thresholds. However, to be able to use the incomplete Cholesky for designing appropriate preconditioner the sparse approximated matrix needs to be positive-definite. For such cases, using the discussion in section 2.4.2.2, $\alpha diag\{diag\{{}^{bd}\tilde{\mathbf{C}}\}\}$ can be added to ${}^{bd}\tilde{\mathbf{C}}$. This makes the matrix diagonally dominant.

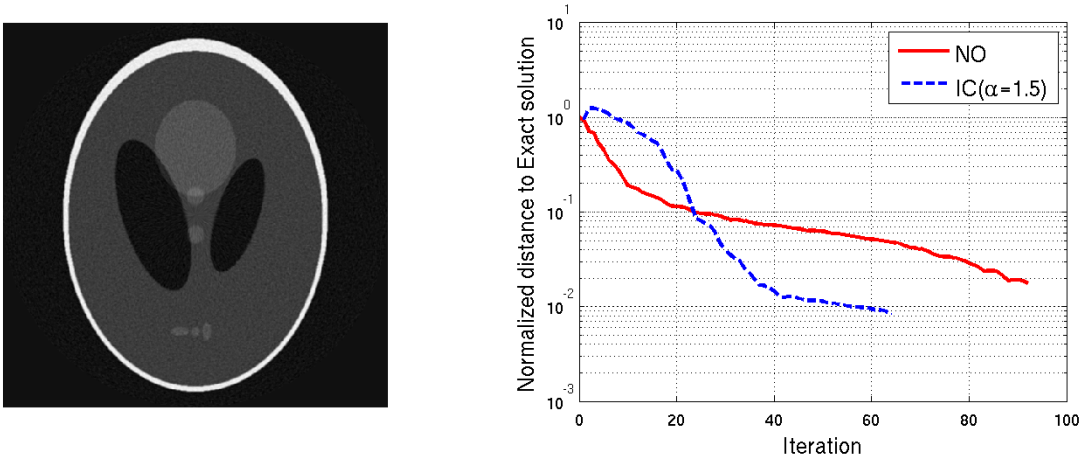
If the approximated normal matrix is close to become nonpositive-definite, although there is mathematically no problem in calculating the incomplete Cholesky factor, the preconditioner will not work efficiently and the convergence can become very slow. One example of such cases is plotted in Fig.(6.10). Here, the regularization type is uniform weighting and the block-wise method is used for the sparse approximation. The desired density level is 0.0078.

By adding $\alpha diag\{diag\{{}^{bd}\tilde{\mathbf{C}}\}\}$, we make an additional approximation but we improve the convergence considerably. This is verified in Fig.6.11, with using $\alpha = 1.5$. Comparing Fig.(6.10a) with Fig.(6.11a), we can see that the circular artifacts are removed and the convergence have been improved effectively.



(a) (b)

Figure 6.10 Effect of having preconditioner,$\mathbf{\Pi}^{-1} = {}^{bd}\tilde{\mathbf{C}}$, with ${}^{bd}\tilde{\mathbf{C}}$ close to becoming nonpositive-definite: (a) Reconstructed image, (b) Convergence.



(a) (b)

Figure 6.11 Correcting the nonpositive-definite problem using α: (a) Reconstructed image, (b) Convergence.

Once the minimum value of α has been determined, it is interesting to see if further increasing the value of α affects the convergence. To do this, the above experiment has been repeated for 5 different values of α. The results are presented in Fig.(6.12). We can see here that increasing α improves the convergence. However, there is a certain limit for increasing α, and for α greater than this limit, no more improvement is observed. This can also be seen in Fig.(6.12) for $\alpha \geq 9$. Consequently, the optimal value of α needs to be found experimentally. It should be noted that the parameter α is dependent on several factors, which include: the type of regularization, regularization parameter (λ), the sparsity level and the image size. Moreove, increasing α has no

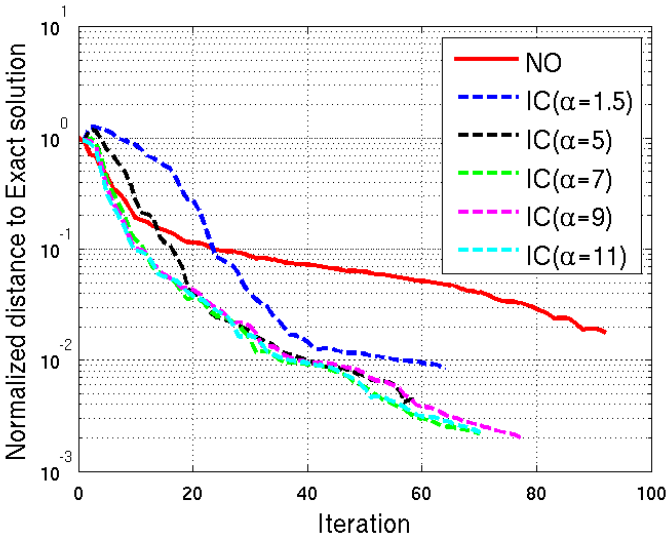


Figure 6.12 Effect of α in performance of preconditioners (incomplete Cholesky decomposition with block-wise sparse approximation).

cost, so it is rational to find the optimal value for this parameter to get the best possible performance of each preconditioner at desired sparsity level. We now provide more details about the proper choice of α for nonuniform and uniform weighting.

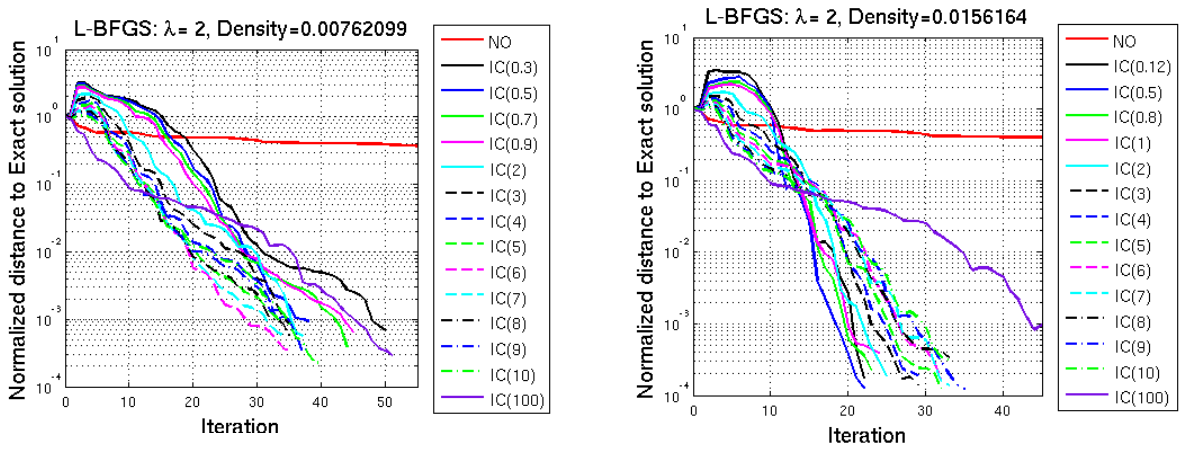
Nonuniform weighting

Fig.(6.13) shows the performance of each preconditioner for different values of α by comparing their normalized distance to the exact solution. The block-wise method has been used for making a sparse approximation to ${}^{bd}C$. We can see that the performance of preconditioners have improved by increasing the value of α. For the density level of 0.0078, the optimal value for α is 6 as no more improvement is observed when this parameter is further increased (6.13a). For the density level=0.0156, Fig.(6.13b) shows the optimal α to be 0.5.

Uniform weighting

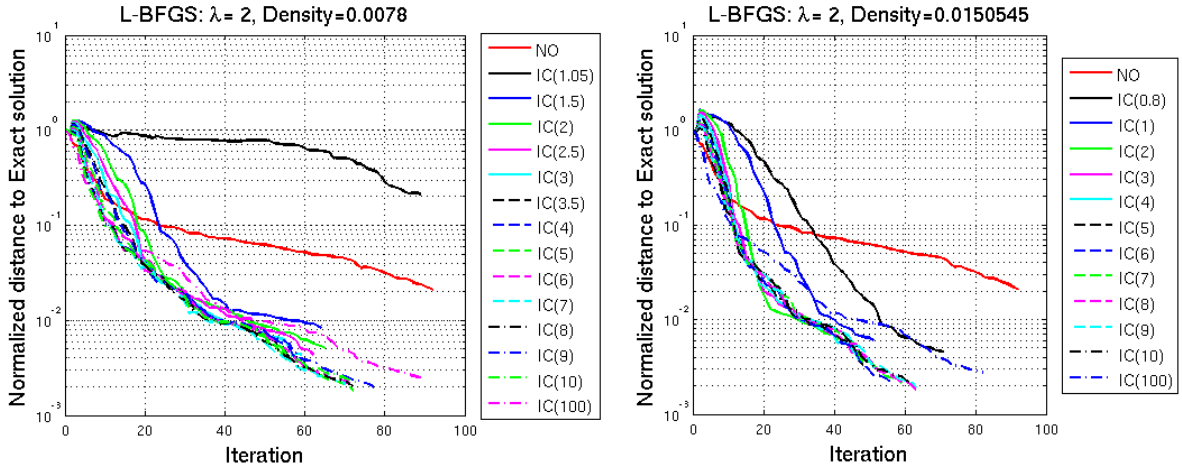
Fig.(6.14a) and (6.14b) show the performance of each preconditioner for different values of α by comparing their normalized distance to the exact solution for density levels 0.0078 and 0.0156, respectively. Similarly to nonuniform weighting, the block-wise method has been used for making the sparse approximations.

Here, the optimal value of α for both sparsity levels is found to be 3,



(a) (b)

Figure 6.13 Nonuniform weighting, Comparison of the performance of preconditioners for different α using block-wise sparse approximation: (a) density=0.0078 , (b) density=0.0156, i.e., the number in the parenthesis in legend is α used for each experiment.



(a) (b)

Figure 6.14 Uniform weighting, Comparison of the performance of preconditioners for different α using block-wise sparse approximation: (a) density= 0.0078, (b) density= 0.0156,i.e., the number in the parenthesis in legend is α used for each experiment.

6.4.2.5 Performance of preconditioners for approach 2

In this section performance of three preconditioners obtained using block-diagonalization of normal matrix (${}^{bd}\mathbf{C}$) will be compared. This include diagonal preconditioner introduced in section 5.4.1 and the two sparse preconditioners using incomplete Cholesky decomposition discussed in section 5.4.2. The number of nonzero elements stored and used for each preconditioner is as following:

- Diagonal Preconditioner: $K \times N = 262,144$ elements, or 2.1 MB
- Sparse approximation, density=0.0078: $2 \times 262,144$ elements, 4.2 MB
- Sparse approximation, density=0.0156: $4 \times 262,144$ elements, 8.4 MB

The parameters which were used for incomplete Cholesky-based preconditioners were found experimentally and are listed in Table 6.3 below:

Table 6.3 Parameters for FFT-based preconditioners (Image size: 512×512, Phantom: Shepp-Logan, Regularization: $L2$, Noise model: Gaussian, SNR= 30dB)

			Regularization	
			Uniform weighting	Nonuniform weighting
♯ nonzero elements stored	$2 \times KN$	δ	1.5×10^{-1}	2×10^{-2}
		α	3	6
	$4 \times KN$	δ	8×10^{-2}	5×10^{-4}
		α	3	0.5

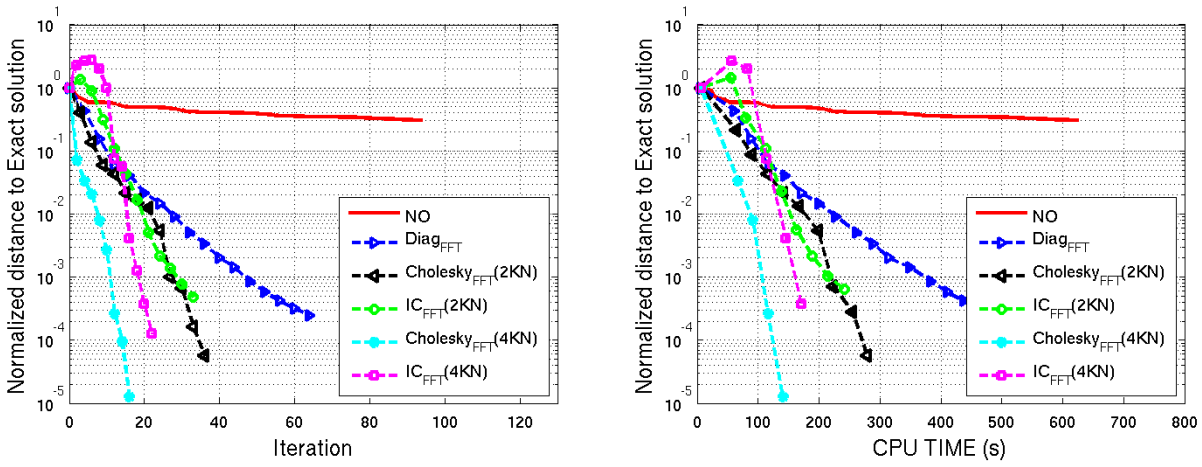
Note again that, based on the results obtained in section 6.4.2.2 block-wise method has been used for making the sparse approximation to ${}^{bd}\mathbf{C}$.

To study the effect of additional approximations made by using incomplete Cholesky decomposition, preconditioner using the exact Cholesky decomposition has also been included in the experiment for each of the sparse preconditioners. These types will never be used in practical use. This is because, as already discussed, exact Cholesky decomposition destroys the sparsity of block ($\mathbf{\Pi}_i$) and has the same cost as saving the complete ${}^{bd}\mathbf{C}$.

Nonuniform weighting

Fig.(6.15) shows the performance of the three preconditioners under investigation and of the exact Cholesky of the sparse preconditioners. From these plots we can see that preconditioners obtained using the FFT method are quite efficient and improve the convergence considerably.

Moreover, in this figure we can see that just as we expected, the performance of preconditioners with exact Cholesky are better than their incomplete Cholesky counterpart. This is reasonable because we are making some approximation by using the incomplete Cholesky decomposition. However, one can also observe that using this approximation, the performance is still good. Considering the gain achieved in terms of memory need, when incomplete Cholesky is used (4.2 and 8.4MB for incomplete Cholesky versus 268MB for exact Cholesky), and the small decrease in convergence speed, this method will be used practically.



(a) (b)

Figure 6.15 Nonuniform regularization, FFT-based preconditioners performance: (a) Convergence, (b) reconstruction time

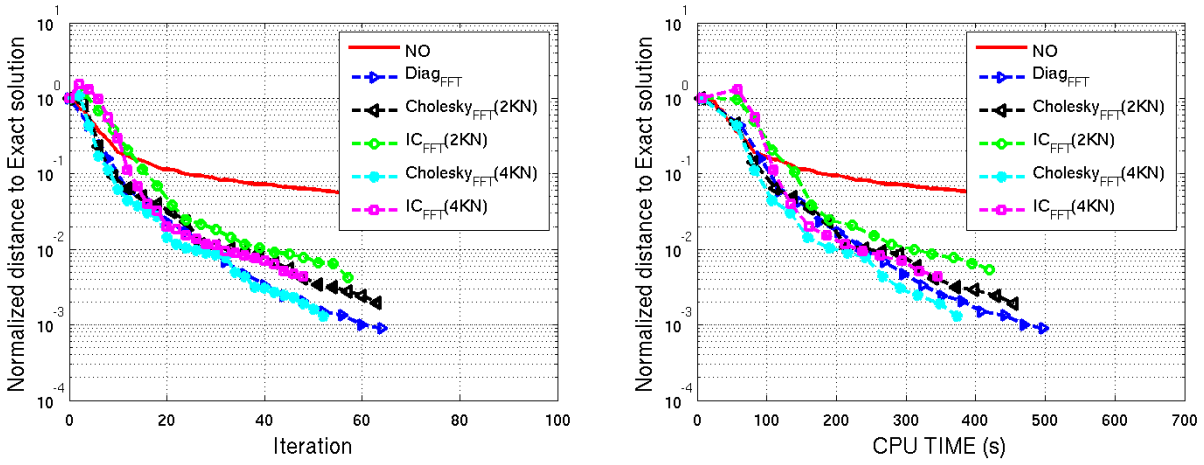
This figure also shows that the performance of the preconditioners increase as their sparsity levels decrease. This can also be mathematically justified as the approximations decrease by increasing the density level. Therefore, the sparsity level can be chosen based on the size of problem, maximum reconstruction time allowed, the available memory and the desired quality of the reconstruction.

Note that the diagonal preconditioner is showing a good performance considering its low memory need. It is also showing a better performance than the sparse preconditioner for first few iteration. For some applications (with loose restrictions on reconstruction time), diagonal preconditioners can be efficient enough.

Uniform weighting

Similarly to nonuniform regularization, performance of the FFT-based preconditioners are compared when uniform weighting is used for regularization. Fig.(6.16a) shows how the normalized distance to the exact solution decreases as a function of the number of iterations and as a function of CPU time for each preconditioner. To study the efficiency of preconditioning for this type of regularization, one can compare the number of iterations required to decrease the normalized distance to 0.08 for each experiment as a typical case. For no preconditioning 80 iterations is required whereas using preconditioner this can reduce to 20 which mean 4-times reduction in time.

It was shown that for uniform weighting the deterioration of the condition number is not as severe as the nonuniform weighting. Therefore, even without using any preconditioners the problem will eventually converge if maximum number of iterations is set to a large value. This means that the benefit of preconditioning for uniform weighting is not as important as with nonuniform



(a) (b)

Figure 6.16 Uniform regularization, FFT-based preconditioners performance: (a) Convergence, (b) reconstruction time

weighting.

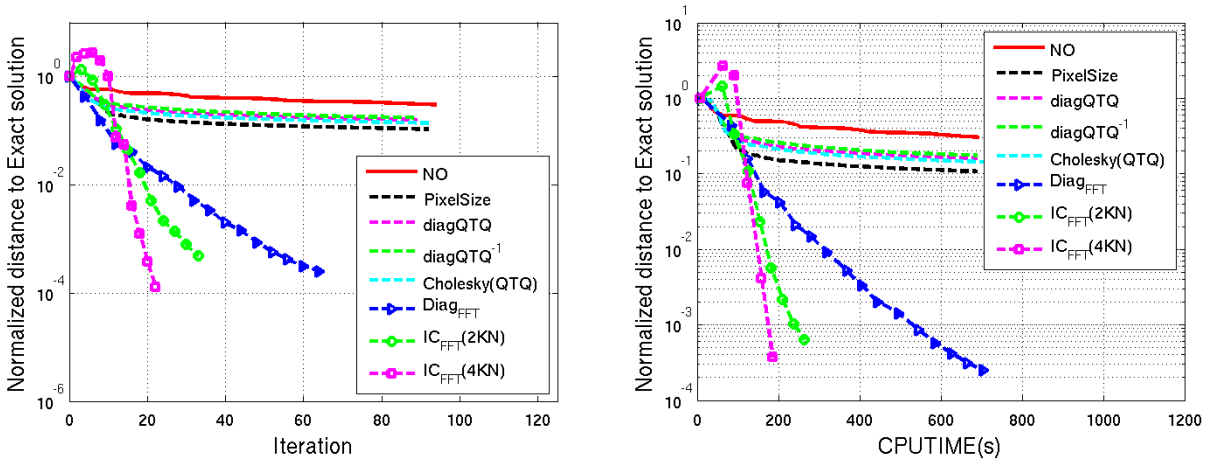
6.4.3 Global comparison of the preconditioners

In this section performance of different preconditioners designed in this work is compared for each type of regularization function used. The effect of preconditioning on the improvement of the eigenvalue spectrum of the normal matrix will then be studied in section 6.4.3.1. The efficiency of the designed preconditioners will then be validated by using different SNR levels and different noise model for the measured data in sections 6.4.3.2 and 6.4.3.3, respectively.

Nonuniform weighting

Fig.(6.17a) shows the performance of 7 different preconditioners when nonuniform regularization has been used. It can be observed that FFT-based preconditioners are performing considerably better than the ones obtained by block-diagonal approximation to the normal matrix. Moreover, we can see that preconditioners of the first approach are not efficient and so the problem does not converge even with the preconditioners used. This can also be seen by comparing the reconstructed images in Fig.(6.19). This is because using nonuniform regularization will further deteriorate the condition number so that a more accurate preconditioner is required to correct the condition number and improve the convergence.

This problem has been well addressed by using FFT-based preconditioners. This is verified in Fig.(6.17) and also on the reconstructed images using such type of preconditioners (Fig.(6.19f), Fig.(6.19g) and Fig.(6.19h). In Fig.(6.17) we can see that using FFT-based preconditioners convergence speed has been improved significantly and the optimization problem converges in about 15



(a) (b)

Figure 6.17 Nonuniform Regularization: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

iterations.

To compare the complexity of each method per iteration, the normalized distance to exact solution is plotted versus CPU time in Fig.(6.17b). Using this figure we can compare the reconstruction time with each type of preconditioner to make a valid conclusion.

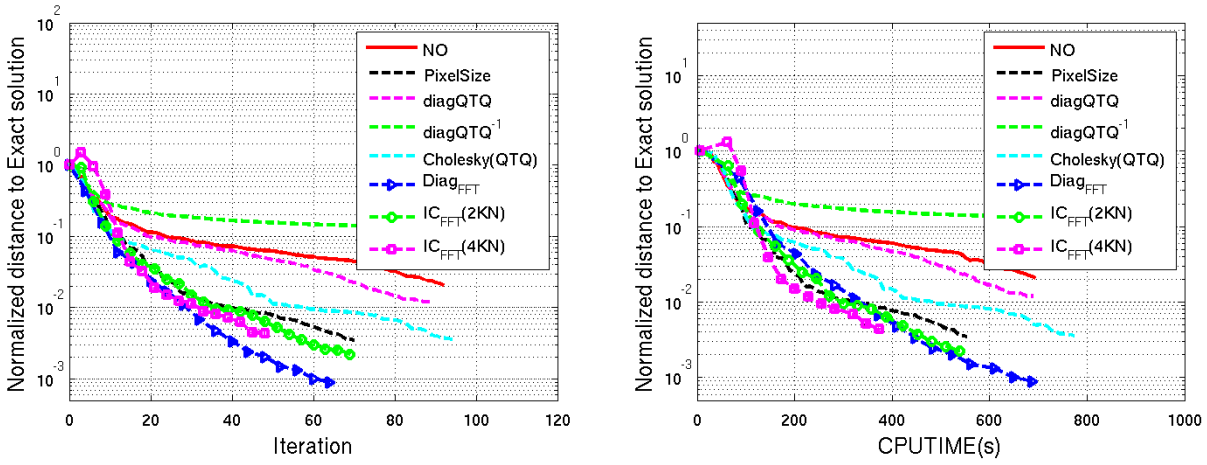
Fig.(6.17b) demonstrates that the additional complexity of the Cholesky preconditioner (for the first approach) and incomplete Cholesky preconditioner (in the second approach) is negligible and that the best trade-off is provided by FFT-based preconditioners.

Uniform weighting

The same experiment was repeated using uniform weighing. Fig.(6.18a) compares the convergence when each of the proposed preconditioners is used.

It can be observed that diagonal preconditioning based on inverse of $\mathbf{Q}_0^T\mathbf{Q}_0$ is not useful. The other preconditioners improve convergence. The improvement is low for preconditioning based on the diagonal values of $\mathbf{Q}_0^T\mathbf{Q}_0$ and Cholesky decomposition of this block, but the preconditioner based on pixel size and the FFT-based preconditioners ones are found to be efficient.

Here, it should be noted that Fig.(6.18a) suggests that all the FFT-based preconditioners are behaving in a very similar manner with the diagonal one being the most efficient. Fig.(6.18b) proves that considering the reconstruction time, improvement in convergence is larger in the sparse incomplete Cholesky preconditioners compared to the diagonal preconditioner with better convergence for the one with higher density level. Overall, from these figures we can conclude that preconditioning based on pixel size is also efficient when using uniform weighting for regularization. If one can pay more in terms of memory, further reduction in reconstruction time can be achieved.



(a) (b)

Figure 6.18 Uniform Regularization: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

The reconstructed images using each of these preconditioners are shown in Fig.(6.20). One can see that preconditioning using the diagonal of the inverse of Υ is not useful as the reconstruction problem has not converged.

As it was discussed in chapter 4, using uniform weighting does not deteriorate the condition number as much as the nonuniform weighting does. Therefore, preconditioning is not as essential as the nonuniform case. This means that the problem may converge even without having preconditioner however using preconditioners, speed of convergence can be further increased.

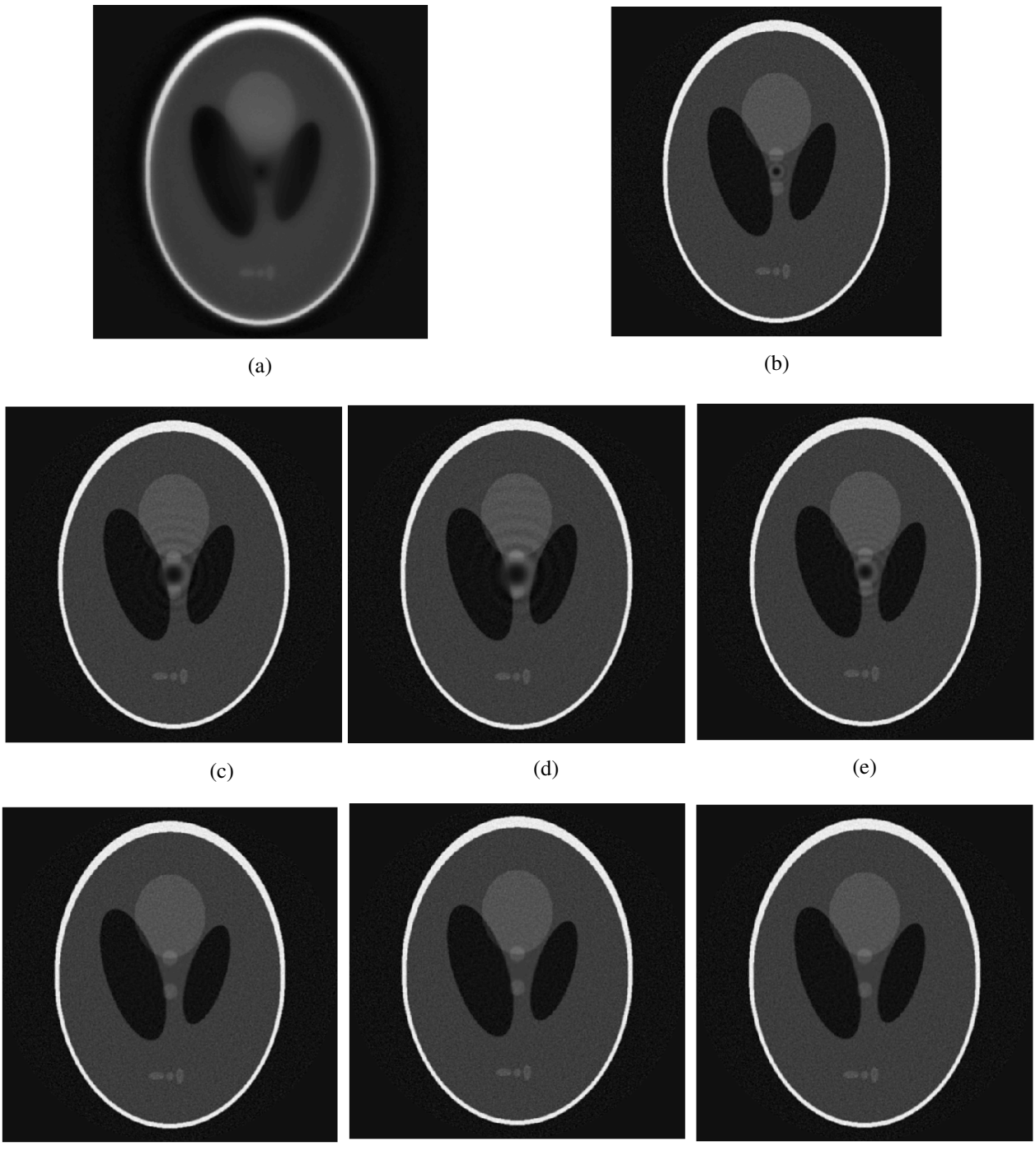
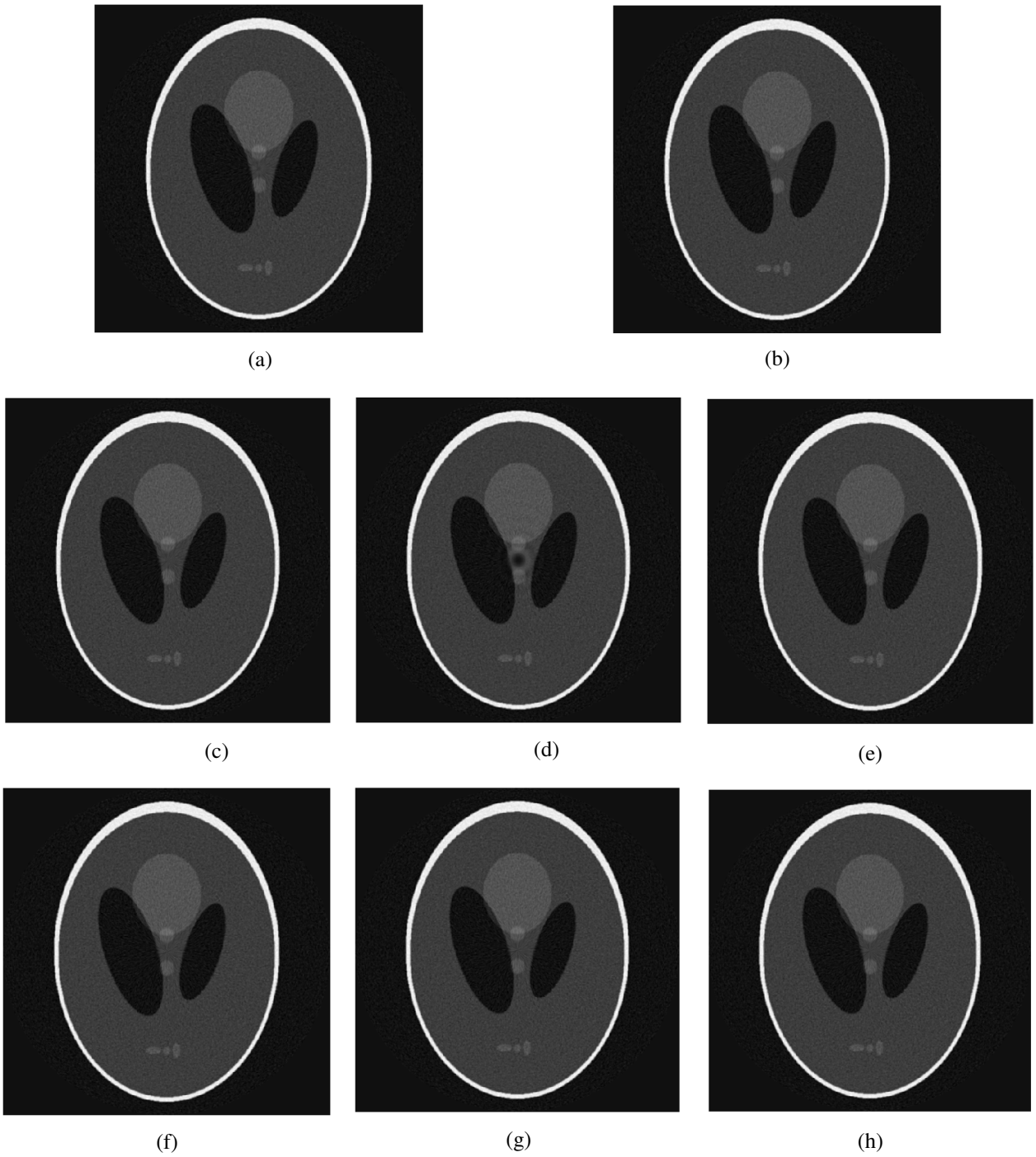


Figure 6.19 Nonuniform regularization-Reconstructed images with: (a) No preconditioner, Preconditioning based on: (b) Pixel sizes, (c) Diagonal of $\mathbf{Q}_0^T\mathbf{Q}_0$, (d) Diagonal of $(\mathbf{Q}_0^T\mathbf{Q}_0)^{-1}$, (e) Cholesky decomposition on $\mathbf{Q}_0^T\mathbf{Q}_0$, (f) Diagonal of ${}^{bd}\mathbf{C}$, (g) Incomplete Cholesky on ${}^{bd}\mathbf{C}$ with density level 0.0078, (h) Incomplete Cholesky on ${}^{bd}\mathbf{C}$ with density level 0.0156.



(a) (b)

(c) (d) (e)

(f) (g) (h)

Figure 6.20 Uniform regularization- Reconstructed images with: (a) No preconditioner, Preconditioning based on: (b) Pixel sizes, (c) Diagonal of $\mathbf{Q}_0^T\mathbf{Q}_0$, (d) Diagonal of $(\mathbf{Q}_0^T\mathbf{Q}_0)^{-1}$, (e) Cholesky decomposition on $\mathbf{Q}_0^T\mathbf{Q}_0$, (f) Diagonal of ${}^{bd}\mathbf{C}$, (g) Incomplete Cholesky on ${}^{bd}\mathbf{C}$ with density level 0.0078, (h) Incomplete Cholesky on ${}^{bd}\mathbf{C}$ with density level 0.0156.

6.4.3.1 Preconditioners and their effect on the eigenvalue spectrum

In the previous section, the performance of the preconditioners were studied by plotting the convergence curves versus iteration number and CPU time and it was concluded the the FFT based preconditioners are efficient in improving the convergence. In this section, the effect of preconditioning and how they affect the eigenvalue spectrum of the normal matrix is studied. To do this the magnitude of the eigenvalues is plotted for the new normal matrix: $\mathbf{\Pi}^T\{\mathbf{P}^T\mathbf{P}+\lambda\mathbf{D}^T\mathbf{D}\}\mathbf{\Pi}$ when each preconditioner is used.

Fig.(6.21a) shows the eigenvalue spectrum for different preconditioners when uniform weighting is used. The efficiency of the preconditioners can be compared by looking at how they are spread in each case. For preconditioners designed using the first approach, Cholesky-based preconditioner has the best performance. The efficiency of FFT-based preconditioners is evident in this figure. This is because as can be seen in Fig.(6.21a) using these preconditioners the range at the which the magnitude of the eigenvalues are spread has significantly decreased. Similarly to what has been observed in previous sections, the diagonal preconditioner from the second approach is performing better than the other preconditioners.

A similar behaviour is seen in Fig.(6.21b) for nonuniform weighting with better performance for FFT-based preconditioners. For this case looking at the eigenvalue spectrum when no preconditioner has been used, we can see that the eigenvalues are spread on a wider range compared to the uniform case. Again, Cholesky-based preconditioner has the best performance among the preconditioners which have been designed using the first approach. This is consistent with the results obtained in previous sections. FFT-based preconditioners show a significant improvement in

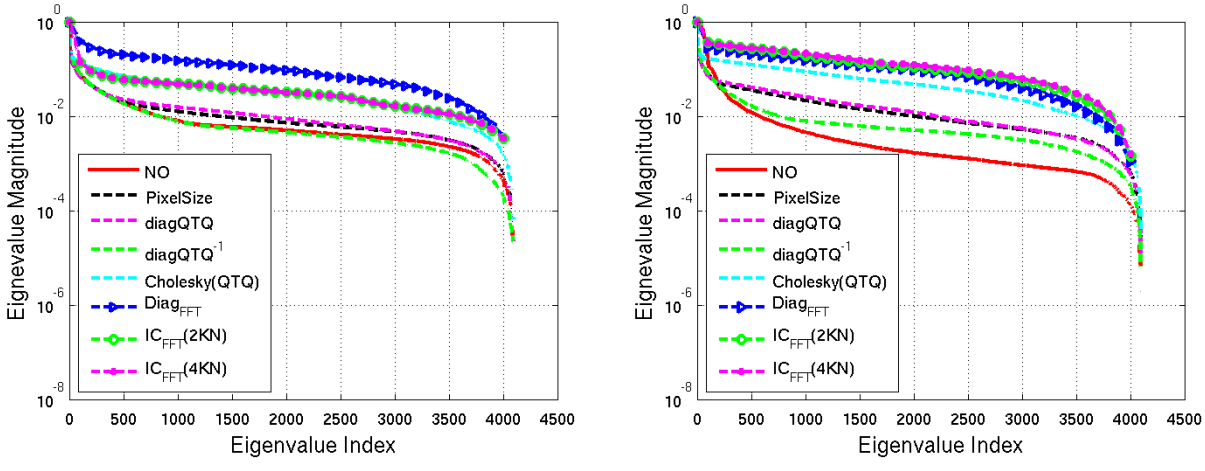


Figure 6.21 Efficiency of preconditioners for improving the eigenvalue spectrum (a) Uniform regularization, (b) Nonuniform regularization (SNR=30dB, $\lambda = 2$).

both decreasing the condition number and the range in which the magnitude of the eigenvalues are spread.

6.4.3.2 Sensitivity to the noise level

To study the performance of these preconditioners for different SNR levels, this experiment was repeated for SNR of 25 and 35 dB. By doing this we can compare if the same pattern of convergence is observed and the preconditioners which were concluded to be efficient for SNR= 30dB remain useful. Making a general conclusion on the obtained results will then be possible. This is done for completeness of the study on the performance of the designed preconditioners.

The optimal values of parameters δ and α were found experimentally and are summarized in Table 6.4.

Table 6.4 Parameters for FFT-based preconditioners (image size: 512×512, phantom: Shepp-Logan, regularization: $L2$, noise model: Gaussian, SNR= 25dB and SNR= 35dB)

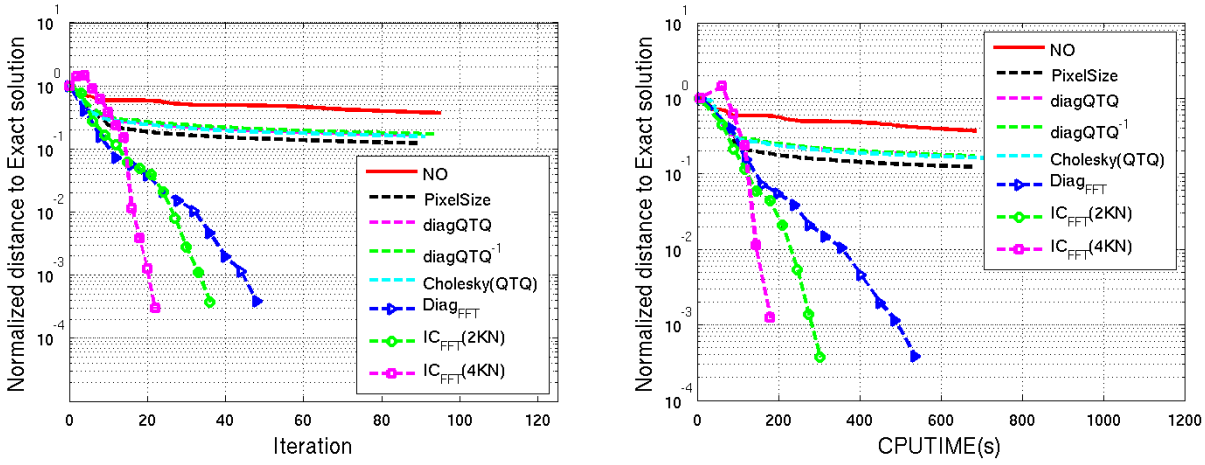
Noise level	PC		Regularization	
			Uniform weighting	Nonuniform weighting
SNR= $25dB$	$2 \times KN$	δ	0.18	0.008
		α	3	6
	$4 \times KN$	δ	0.07	0.0002
		α	2	1
SNR= $35dB$	$2 \times KN$	δ	0.01	0.04
		α	3	6
	$4 \times KN$	δ	0.006	0.0005
		α	2	3

Nonuniform weighting

Fig(6.22) shows the performance of preconditioners with nonuniform regularization and SNR= 25dB. Comparision with Fig.(6.17) shows that the convergence patterns are similar for the two SNR values. It should be noted here that although the noise level has increased FFT-based preconditioners improve convergence significantly, and the number of iterations required to converge remain comparable to those of the 30dB case. A similar behavior is seen for SNR= 35dB in Fig.(6.23).

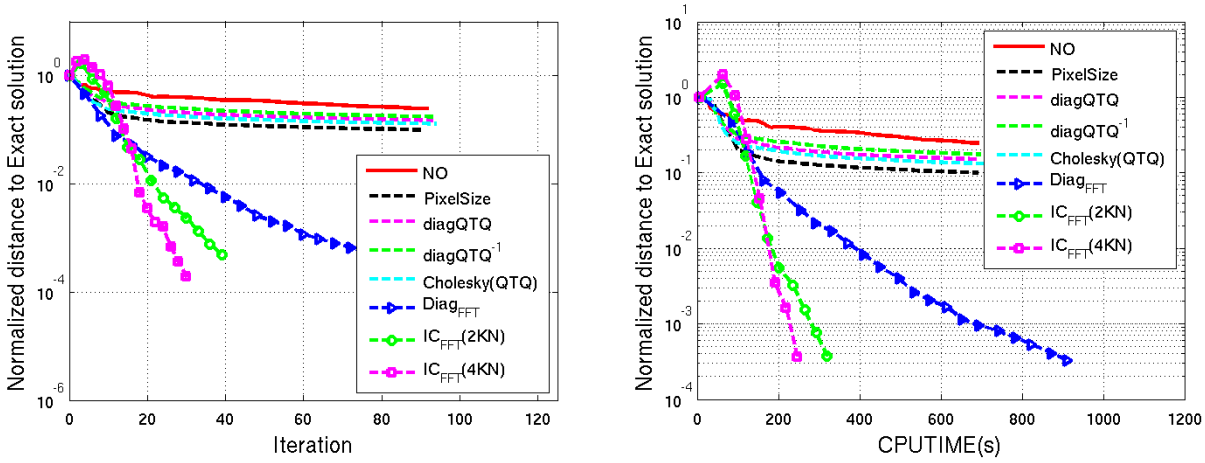
Uniform weighting

Fig(6.24) shows the performance of preconditioners when uniform regularization have been used and SNR= 25dB. The pattern of convergence for preconditioners are similar to those seen for SNR= 30dB in Fig.(6.18). Again, we can see that the preconditioner based on pixel size from the first approach has a good performance when uniform regularization have been used. Indeed,



(a) (b)

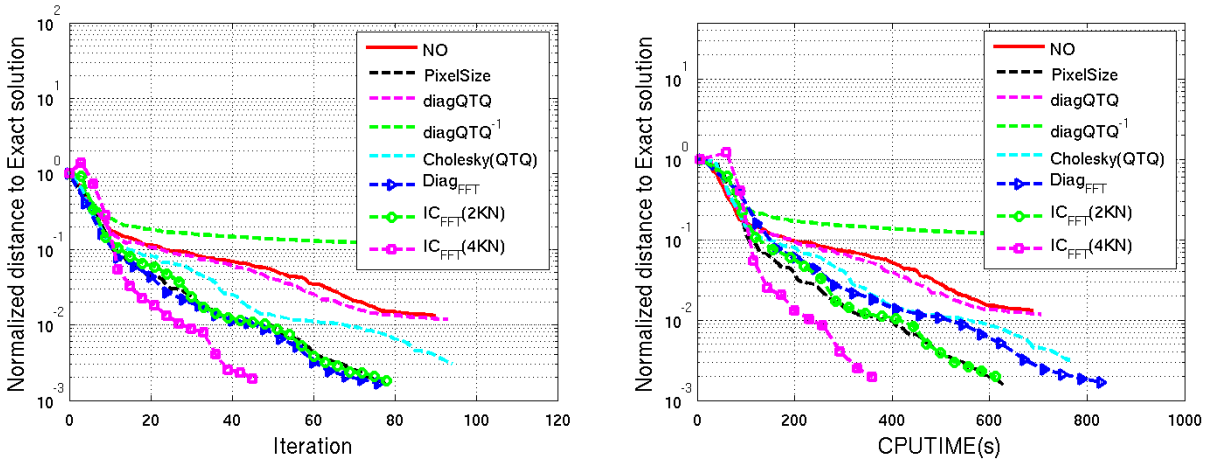
Figure 6.22 Nonuniform Regularization, SNR=25dB: Performance of preconditioners, (a) Convergence, (b) reconstruction time.



(a) (b)

Figure 6.23 Nonuniform Regularization, SNR=35dB: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

in Fig.(6.24b), one can see that the performance of this preconditioner is better than that of diagonal preconditioning in the Fourier domain and is very similar to that of incomplete Cholesky decomposition with $2KN$ nonzero elements. The fastest convergence corresponds to the incomplete Cholesky decomposition with the largest number of non-zero elements.

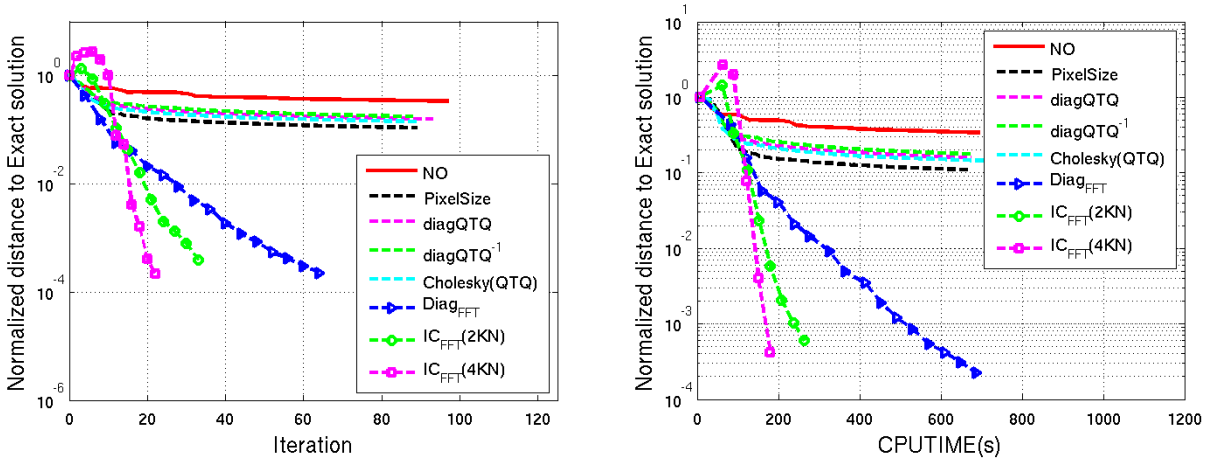


(a) (b)

Figure 6.24 Uniform Regularization, SNR=25dB: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

6.4.3.3 Sensitivity to the noise model

Considering the quantum nature of photons, the noise which is usually added to the sinogram is a mix of Gaussian and Poisson noise. Here, to study if these preconditioners remain efficient when the noise added to the sinogram has Gaussian distribution with variance dependent on the photon counts, the experiment is repeated when this model is used for the additive noise. Note that here, $\mathbf{\Sigma}$,

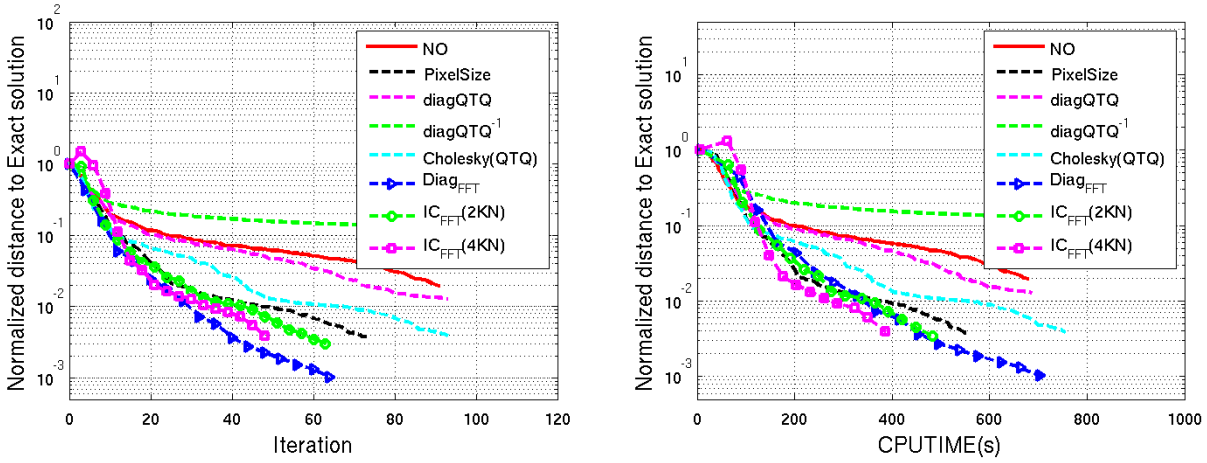


(a) (b)

Figure 6.25 Nonuniform Regularization, SNR=30dB: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

the matrix which appears in (3.1) was set to identity; this allows using the preconditioners designed in this work.

Fig.(6.26) and Fig.(6.25) show the performance of preconditioners when nonuniform and uniform regularization is used respectively. From these figures we can see that the convergence patterns are similar to what was observed when the additive noise was Gaussian. This proves that the designed preconditioners are efficient for both models of noise added to the projection data.



(a) (b)

Figure 6.26 Uniform Regularization, SNR=30dB: Performance of preconditioners, (a) Convergence, (b) reconstruction time.

6.4.4 Image quality: uniform or nonuniform weighting?

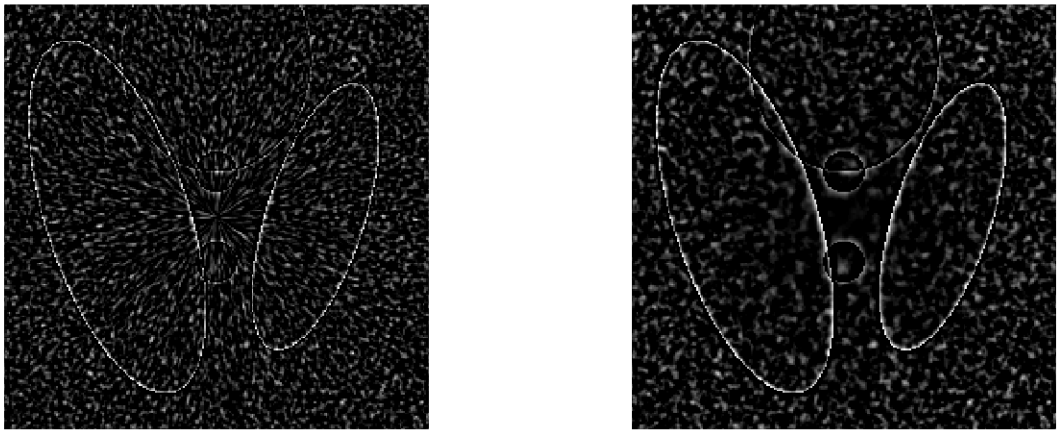
In this section, the quality of images reconstructed using either type of penalty function will be compared. The best preconditioner which was designed and studied in the previous section will be used for comparing the images (Fig.(6.28)). In both methods, it was seen that FFT-based incomplete Cholesky decomposition with the highest number of elements overtake the rest in terms of number of iterations required to get to the same distance to exact solution.

Now, to compare the quality of reconstructed images when each type of regularization functions is used, we take a closer look at the center of reconstructed images where convergence is the slowest in polar discretization. Fig.(6.28) shows the difference between the recosntructed images and the actual phantom. One can see in this figure that for uniform weighting, there are some streak artifacts in the center. This can be attributed to the approximation to the gradient which has been made by using uniform weighting. These artifacts are removed when nonuniform weighing has been used(Fig.(6.28b)).

One quantitative indicator which can be used for comparing the quality of reconstructed images is the root mean square error (RMSE), which is the quadratic norm of the difference the reconstructed image and the actual phantom. It can be expressed as follows:

(a) (b)

Figure 6.27 Reconstructions obtained :(a) uniform weighting, (b) nonuniform weighting (SNR=30dB).



(a) (b)

Figure 6.28 Comparison of image quality for:(a) uniform weighting, (b) nonuniform weighting (SNR=30dB).

$$RMSE = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mu(i) - \mu_{true}(i))^2} \tag{6.2}$$

where μ_{true} represents the actual phantom. This is calculated for both uniform and nonuniform regularization functions. The RMSE is found to be 0.0125 and 0.0144. This shows that the error

is very similar for both types of weighting with a slight advantage for the uniform case. However it should be noted that the RMSE does not fully capture the complexity of assessing the quality of reconstructions. This is evident by looking at the reconstructed images in Fig.(6.28). Therefore, further study of the performance of this type of discretization and more complete validation of this technique are required.

6.5 Conclusion

In this chapter it was seen that although using polar grid discretization deteriorates the condition number of the normal matrix, this problem can be addressed by designing appropriate preconditioners.

Using the uniform weighting in regularization function, the deterioration of the condition number is not as severe as for nonuniform weighting. However, the use of nonuniform weighting is consistent with the definition of penalty term and so is a better numerical approximation to the gradient. In this work, the eigenvalue spectrum of the normal matrix for both types of weighting were improved by designing efficient preconditioners. This resulted in faster convergence of iterative methods.

In section 6.4.3.2, consistency of results was proved for different SNR levels and section 6.4.3.3 shows that the preconditioners remain efficient when the noise in the projection data has the Poisson distibution. Therefore it is possible to list the preconditioners based on their performance from the most efficient to the least.

Using numerical simulations, it was observed that FFT-based preconditioners show a significant improvement for both type of regularizations compared to those designed in spatial domain. Using preconditioners from the incomplete Cholesky decomposition in the Fourier domain, a further improvement can be achieved at the cost of a larger memory footprint.

Quantitative indicators used in section 6.4.4 show that despite having lower distance to the exact solution for nonuniform case RMSE is slightly smaller for uniform regularization. Performance and efficiency of this technique needs to be further explored in future works.

CHAPTER 7

CONCLUSION

7.1 Summary of work

The objective of this work was to develop efficient tools which allow reconstruction of medical size images in X-ray tomography. These tools were developed in the framework of solving an optimization problem using iterative methods for reconstructing the object.

To do this, polar coordinates is used for discretizing the object. Using this type of representation for the object introduces a large redundancy in the coefficients of projection matrix and leads to a significant reduction in the memory need. However using this type of representation raises several questions which include: possibility of efficient implementation of projection and backprojection operations, conditioning of reconstruction problem, derivation of penalty functions consistent with the polar discretization scheme and actual quality of reconstructions results. These questions were addressed in this work.

This was done by using a simple 2D framework, a quadratic (L_2) penalty term and a Gaussian noise model with uniform variance ($\mathbf{\Sigma} = \mathbf{I}$).

In chapter 4 it was shown that when the object is discretized in polar coordinates, the projection matrix ($\mathbf{P}$) has a BC structure and can be represented by a partial projection operation limited to one projection angle. This property yields easy parallel implementation of the projection operation by using a circular shift on the object and by treating each projection independently. In addition this property can be further used to obtain a BC structure for the $\mathbf{P}^T$ which can be exploited for parallelizing the backprojection operation. Therefore, it was proved that using polar discretization, the projection and backprojection operations can be implemented efficiently using parallelization techniques.

Using polar tessellation for discretizing the object introduces size-varying pixels which will affect the conditioning of the reconstruction problem and decreases the convergence speed of numerical methods. Deterioration of the the condition number of reconstruction problem becomes more severe when nonuniform weighting is used for the regularization function.

To address the conditioning problem discussed above, several preconditioners were designed using the BC structure of the normal matrix. These preconditioners were designed using two different approaches, one by making a block-diagonal approximation to the normal matrix and one by block diagonalizing the normal matrix using FFT techniques. In designing these preconditioners, it was seen that there is always a trade-off between the efficiency of preconditioners in terms of

increase in the convergence speed and the memory requirement. Diagonal and Cholesky-based preconditioners were designed for each case and their efficiency was tested using several experiments with realistic data size. Different SNRs and noise models were used for projection data to evaluate the performance of these preconditioners. The results in chapter 6 verified their efficiency and showed that FFT-based preconditioners perform significantly better in increasing the convergence speed.

For comparing the quality of reconstruction using uniform and nonuniform weighting, we used the reconstructed images obtained with the best performing preconditioner designed in this work and both types of regularization functions. The RMSE of these reconstructed images was then calculated for each case. Results show that the root mean square in both methods are similar however visually the one using nonuniform weighting is showing a better quality. A more accurate indicator which can captures the complexity of assessing the quality of reconstructed image needs to be found for this type of comparison.

To conclude, it was seen that representing the object in a polar coordinate system for 2D CT reconstruction presents appealing characteristics: low memory requirements, simple structure, easy computation of the projection operator, straightforward parallelization of the projection and back-projection operations. Using appropriate preconditioners the convergence speed of the numerical methods can be sufficiently increased. Therefore, the polar representation can be viewed as an interesting alternative to standard Cartesian representations.

7.2 Limitation of the proposed solution

In this work noise was modeled as Gaussian with uniform variance by setting $\mathbf{\Sigma}_b = \mathbf{I}$ in the objective function. The normal matrix has a BC structure for this model of noise. This property has been used in designing preconditioners. However, according to the work of (Sauer et Bouman, 1993), a better model for the quantum noise is to use Poisson noise and choose $\mathbf{\Sigma}_b$ as shown in (2.7). Using this model for $\mathbf{\Sigma}_b$ will destroy the BC property as the noise variance of photons become non-uniform. To overcome this, one alternative is to use some approximation to $\mathbf{\Sigma}_b$ in (2.7) by averaging. Assume that $\mathbf{\Sigma}_b$ can be written as a block-diagonal matrix with diagonal blocks equal to $\mathbf{\Sigma}_b^i$, where $1 \leq i \leq N$ represents the block number, then we can find $\mathbf{\Sigma}_x$ with identical diagnoal blocks as below:

$$\mathbf{\Sigma}_x = \begin{bmatrix} \mathbf{\Sigma}_a & & & \\ & \mathbf{\Sigma}_a & & \\ & & \ddots & \\ & & & \mathbf{\Sigma}_a \end{bmatrix} \tag{7.1a}$$

and

$$\mathbf{\Sigma}_a(j,j) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{\Sigma}_b^i(j,j), \qquad j = 1,2,\ldots,K \tag{7.1b}$$

This new $\mathbf{\Sigma}$ will preserve the BC property of the normal matrix as all the diagonal blocks are identical. The efficiency of this algorithm needs to be tested experimentally.

7.3 Proposed future work

The results obtained in this work can be used to develop an algorithm for 3D reconstruction of tomography images. Cylindrical coordinates can be used for this purpose. The efficiency of the techniques developed in this work will be more evident when used for 3D reconstruction.

The current work can also be used and expanded to the polychromatic model. It is known that for either of the models, polychromatic or monochromatic, the key problem which is performing the projection and backprojection at every iteration, remains the same. Therefore, the algorithm developed in this work to parallelize these operations efficiently can be taken advantage of and used with the polychromatic model.

Another issues which remains to be studied as future work is how to adopt this work to non-quadratic regularization functions. Using L_2L_1 as an example of this family of regularization, one can highly benefit from their edge-preserving property.

REFERENCES

AHN, S., FESSLER, J. A., BLATT, D. et HERO, III, A. O. (2006). Convergent incremental optimization transfer algorithms: Application to tomography. 25, 283–296.

BERTSEKAS, D. P. (1999). Nonlinear programming.

BOUMAN, C. A. et SAUER, K. D. (1993a). A generalized Gaussian image model for edge-preserving MAP estimation. IP-2, 296–310.

BOUMAN, C. A. et SAUER, K. D. (1993b). A local update strategy for iterative reconstruction from projections. 41, 534–548.

BOUMAN, C. A. et SAUER, K. D. (1996). A unified approach to statistical tomography using coordinate descent optimization. 5, 480–492.

BRANKOV, J. G., YANG, Y. et WERNICK, M. N. (2004). Tomographic image reconstruction based on a content-adaptive mesh model. 23, 202–212.

CHEN, K. (2005). *Matrix preconditioning techniques and applications*, vol. 19. Cambridge University Press.

DE MAN, B. et BASU, S. (2002). Distance-driven projection and backprojection. *Nuclear Science Symposium Conference Record*. IEEE, vol. 3, 1477–1480.

DE MAN, B. et BASU, S. (2004). Distance-driven projection and backprojection in three dimension. 49, 2463–2475.

DE MAN, B., BASU, S., THIBAULT, J.-B., HSIEH, J., FESSLER, J. A., BOUMAN, C. et SAUER, K. (2005). A study of four minimization approaches for iterative reconstruction in X-ray CT. *2005 IEEE Nuclear Science Symposium Conference Record*. 2708–2710.

DE MAN, B., NUYTS, J., DUPONT, P., MARCHAL, G. et SUETENS, P. (2000). Reduction of metal streak artifacts in X-ray computed tomography using a transmission maximum a posteriori algorithm. 47, 977–981.

DE MAN, B., NUYTS, J., DUPONT, P. et SUETENS, P. (2001). An iterative maximum-likelihood polychromatic algorithm for CT. 20, 999–1008.

ELBAKRI, I. A. et FESSLER, J. A. (2002). Statistical image reconstruction for polyenergetic X-ray computed tomography. 21, 89–99.

ERDOĞAN, H. et FESSLER, J. A. (1999a). Monotonic algorithms for transmission tomography. 18, 801–814.

ERDOĞAN, H. et FESSLER, J. A. (1999b). Ordered subsets algorithms for transmission tomography. 44, 2835–2851.

FAYYAZI, A. H., HUGATE, R. R., PENNYPACKER, J., GELB, D. E. et LUDWIG, S. C. (2004). Accuracy of computed tomography in assessing thoracic pedicle screw malposition. *Journal of spinal disorders & techniques*, 17, 367–371.

FESSLER, J. A. (1994). Penalized weighted least-squares image reconstruction for positron emission tomography. 13, 290–300.

FESSLER, J. A. (2000). Statistical image reconstruction methods for transmission tomography. J. M. Fritzpatrick et M. Sonka, éditeurs, *Handbook of Medical Imaging*, SPIE Press, Bellingham, WA, vol. 2, chapitre 1. 1–70.

FESSLER, J. A. et BOOTH, S. D. (1999). Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction. 8, 688–699.

FESSLER, J. A., FICARO, E. P., CLINTHORNE, N. H. et LANGE, K. (1997). Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction. 16, 166–175.

GENDRON, D., GOUSSARD, Y., BEAUDOIN, G., SOULEZ, G. et CLOUTIER, G. (2008). Reconstruction of real tomographic data using algebraic methods. Vancouver, BC, Canada, 2717–2720.

GOLDMAN, L. W. (2007). Principles of CT and CT technology. *Journal of nuclear medicine technology*, 35, 115–128.

GOUSSARD, Y., GOLKAR, M., WAGNER, A. et VOORONS, M. (2013). Cylindrical coordinate representation for statistical 3D ct reconstruction. *Proc. Int. Meeting on Fully 3D Image Reconst.* California, U.S.A, 100.

GUAN, H. et GORDON, R. (1996). Computed tomography using algebraic reconstruction techniques (ARTs) with different projection access schemes: a comparison study under practical situations. 41, 1727–1743.

HAMELIN, B., GOUSSARD, Y. et DUSSAULT, J.-P. (2010a). Comparison of optimization techniques for regularized statistical reconstruction in X-ray tomography. *Proc. Int. Conf. Image Proc. Theory, Tools and Appl.* Paris, France. 5 pages.

HAMELIN, B., GOUSSARD, Y., DUSSAULT, J.-P., CLOUTIER, G., BEAUDOIN, G. et SOULEZ, G. (2010b). Design of iterative ROI transmission tomography reconstruction procedures and image quality analysis. 37, 4577–4589.

HAMELIN, B., GOUSSARD, Y., GENDRON, D., DUSSAULT, J.-P., CLOUTIER, G., BEAUDOIN, G. et SOULEZ, G. (2008). Iterative CT reconstruction of real data with metal artifact reduction. 1453–1456.

JIAN, L., LITAO, L., PENG, C., QI, S. et ZHIFANG, W. (2007). Rotating polar-coordinate ART applied in industrial CT image reconstruction. *NDT & E International*, 40, 333–336.

KAUFMAN, L. (1987). Implementing and accelerating the em algorithm for positron emission tomography. 6, 37–51.

KOSTER, J. (2002). Parallel templates for numerical linear algebra, a high-performance computation library.

LEROUX, J.-D., SELIVANOV, V., FONTAINE, R. et LECOMTE, R. (2007). Accelerated iterative image reconstruction methods based on block-circulant system matrix derived from a cylindrical image representation. *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE*. IEEE, vol. 4, 2764–2771.

MC KINNON, G. C. et BATES, R. H. T. (1981). Towards imaging the beating heart usefully with a conventional CT scanner. 28, 123–127.

MENVIELLE, N. (2004). *Réduction des Artéfacts Métalliques en Tomographie à Rayons X*. Mémoire de maîtrise, École Polytechnique, Montréal, Canada.

MORA, C. et RAFECAS, M. (2006). Polar pixels for high resolution small animal PET. IEEE, San Diego, CA, vol. 5, 2812–2817.

MORA, C., RODRÍGUEZ-ÁLVAREZ, M. J. et ROMERO, J. V. (2008). New pixellation scheme for CT algebraic reconstruction to exploit matrix symmetries. 56, 715–726.

MUDRY, K. M., PLONSEY, R. et BRONZINO, J. D. (2003). *Biomedical imaging*. CRC PressI Llc.

MUELLER, J. L. et SILTANEN, S. (2012). *Linear and nonlinear inverse problems with practical applications*, vol. 10. SIAM.

NOCEDAL, J. et WRIGHT, S. J. (1999). *Numerical Optimization*. Operations Research. Springer Verlag, New York, NY.

PRESS, W. H., TEUKOLSKY, S., VETTERLING, W. et FLANNERY, B. (1992). Numerical recipes in c: the art of scientific computing, 994.

QI, J. et LEAHY, R. M. (2006). Iterative reconstruction techniques in emission computed tomography. 51, R541–R578.

RAMANI, S. et FESSLER, J. (2011). Convergent iterative ct reconstruction with sparsity-based regularization. *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*. 302–5.

RAO, G., BRODKE, D., RONDINA, M., BACCHUS, K. et DAILEY, A. (2003). Inter-and intraobserver reliability of computed tomography in assessment of thoracic pedicle screw placement. *Spine*, 28, 2527.

RODRIGUEZ-ALVAREZ, M.-J., SANCHEZ, F., SORIANO, A., IBORRA, A. et MORA, C. (2011). Exploiting symmetries for weight matrix design in CT imaging. 54, 1655–1664.

SANDERSON, J. G. (1979). Reconstruction of fuel pin bundles by a maximum entropy method. 26, 2685–2686.

SANTOS, E., LEDONIO, C. et CASTRO, C. (2011). The accuracy of intraoperative O-arm images for the assessment of pedicle. *Spine*.

SAUER, K. D. et BOUMAN, C. A. (1993). A local update strategy for iterative reconstruction from projections. SP-41, 534–548.

THIBAUDEAU, C., LEROUX, J., PRATTE, J., FONTAINE, R. et LECOMTE, R. (2011). Cylindrical and spherical ray-tracing for ct iterative reconstruction. *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*. IEEE, 4378–4381.

TIKHONOV, A. N. et ARSENIN, V. Y. (1977). *Solutions of Ill-Posed Problems*. Winston, Washington DC.

WANG, G., VANNIER, M., CHENG, P. *ET AL.* (1999). Iterative x-ray cone-beam tomography for metal artifact reduction and local region reconstruction. *Microscopy and Microanalysis*, 5, 58–65.

ZENG, G. et GULLBERG, G. (1992). Frequency domain implementation of the three-dimensional geometric point response correction in SPECT imaging. *IEEE Transactions on Nuclear Science*, 39, 1444–1453.

ZENG, G. L. et GULLBERG, G. T. (1993). A ray-driven backprojector for backprojection filtering and filtered backprojection algorithms. San Francisco, CA, 1199–1201.

ZENG, G. L., HSIEH, Y.-L. et GULLBERG, G. T. (1994). A rotating and warping projector/backprojector for fan-beam and cone-beam iterative algorithm. 41, 2807–2811.

ZHENG, J., SAQUIB, S. S., SAUER, K. et BOUMAN, C. A. (2000). Parallelizable bayesian tomography algorithms with rapid, guaranteed convergence. 9, 1745–1759.

ZHU, C., BYRD, R. H., LU, P. et NOCÉDAL, J. (1997). Algorithm 778. L-BFGS-B: Fortran subroutines for Large-Scale bound constrained optimization. 23, 550–560.

ZHUANG, W., GOPAL, S. S. et HEBERT, T. J. (1994). Numerical evaluation of methods for computing tomographic projections. 41, 1660–1665.

APPENDIX A

METHODS FOR CALCULATION OF THE SYSTEM MATRIX

Pixel-driven, ray-driven and distance-driven are three methods used for generating the projection and backprojection operators. Below a brief explanation on how each method works and its advantages and disadvantages are provided:

Backprojection operator is generated in pixel-driven method by connecting the line integral from the focal spot of source to detector such that it passes through the center of the pixel of interest. The pixel value will then be calculated by interpolation and applying some weighting factors. The limiting factor in this method is that, detectors need to be regularly spaced. Moreover, the high computation complexity involved in this method, requires special hardware implementation and consequently in general purpose microprocessors results in poor performance (De Man et Basu, 2004).

In Ray-driven method, projection operation is calculated by connecting a line from source to center of detector of interest. Image is then reconstructed by finding the contribution of each ray line with each pixel. This method is easy to be used however, suffers from introducing artifact in backprojection and having non-sequential memory pattern access (Zeng et Gullberg, 1993; Zhuang *et al*., 1994).

The artifacts induced by both methods is due to the numerical operations and approximations involved in implementing the forward and back projections and is more crucial in iterative methods as it involves repetition of these operations in every iteration. The accuracy of these methods can be improved by choosing a good interpolation technique or by increasing the number of ray-paths traced per projection beam, dividing each pixel to a number of smaller sub-pixel and then projecting each sub-pixel, and by using circular projections instead of square based projection pixels (Zhuang *et al*., 1994).

(De Man et Basu, 2002) introduced distance-driven method for 2D image reconstruction. This method works by mapping the detector array and the image row/column onto each other along the directions of a common line and then performs a kernel operation over all boundaries. Here the main loop is over the common line of interception rather than the image pixels or detector cells, which makes the calculations fast. (De Man et Basu, 2002) also showed that this method reconcile the advantages of the two common methods, sequential memory access of pixel-driven and low arithmetic cost of ray-driven which makes it suitable for hardware implementation, and avoids the artifacts-inducing approximation made by them however, it is more complex. This is because, in this method the exact length of overlap is used instead of regular interpolation kernel used in the

other two methods.

APPENDIX B

SOME NUMERICAL METHODS FOR IMAGE RECONSTRUCTION

B.1 Gradient-based Methods

Gradient-based methods are of interest for solving large problems because of their linear convergence property and being able to be adapted to nonlinear optimization problems. In this methods, Given a starting point $x_0 \in \mathbb{R}^n$, each iteration follows the relationship below:

$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{D}_k \nabla f(\mathbf{x}_k)} \tag{B.1}$$

Here, $\mathbf{D}_k$ is a symmetric positive-definite matrix and α, a positive scalar, is called the step length. Defining the vector of direction to be $\mathbf{p}_k = \mathbf{D}_k \nabla f(\mathbf{x}_k)$, a decent direction is the one which satisfies $\nabla f(\mathbf{x}_k)^T \mathbf{p}_k < 0$.

B.1.1 Steepest descent

The most obvious choice for search direction is to choose the $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$. This simply means:

$$\mathbf{D}_k = \mathbf{I}, \quad k = 0, 1, \ldots,$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. Equation(B.1) will then reduce to:

$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \nabla f(\mathbf{x}_k)} \tag{B.2}$$

The advantage of this method is that it only needs to calculate the gradient $\nabla f(\mathbf{x}_k)$ and not the second derivatives.

The main disadvantage of steepest descent method is that it can be extremely slow in solving hard problems. Fig(B.1) is taken from the Chapter 3 of (Nocedal et Wright, 1999). One can see from this figure that using this method, problem iterates toward solution in a zigzag format.

This is because according to Theorem 3.3 of (Nocedal et Wright, 1999), as the condition number (the ratio of the largest eigenvalue to the smallest eigenvalue) increases, the contours of the cost function becomes "elongated". Knowing that the gradient direction is almost orthogonal to direction that leads to the minimum, this will cause the zigzag behavior of iterations to be more evident and consequently decreases the convergence.

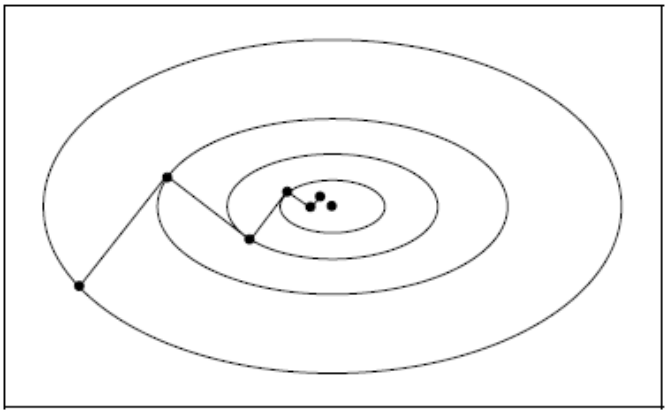


Figure B.1 Steepest descent steps, ©(Nocedal et Wright, 1999)

B.1.2 Linear Conjugate Gradient

This method is used for solving problems with positive-definite coefficients. It is known to be an alternative to Gaussian elimination for large size problems and performs faster than the steepest descent method. The main advantage of this method is that no matrix storage is involved in performing iterations.

The name comes from the fact that a set of conjugate vectors are used as direction of descent, for obtaining the solution. The special property of this method is that one only needs the previous vector p_{k-1} to compute a new vector p_k and each vector will be automatically conjugate to all previous vectors. The algorithm for computing the solution using this method is given below (Nocedal et Wright, 1999):

Algorithm B.1.2

For a linear system of equations $\mathbf{Ax} = \mathbf{b}$
Choose an initial value $\mathbf{x}_0$,
Calculate the residual: $\mathbf{r}_0 = \mathbf{Ax}_0 - \mathbf{b}$, $\mathbf{p}_0 = -\mathbf{r}_0$, $k = 0$;
while $\mathbf{r}_k \neq 0$

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k$$
$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$$
$$\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$$
$$k = k + 1$$

end(while)

One should note that for large size problems calculating the exact minimum is sometimes impractical and some stopping tolerance will be chosen for the algorithm to terminate. The general

idea of CG has been discussed here, one can refer to pg. 169, Chapter 7 of (Nocedal et Wright, 1999) for the extension of this algorithm for large size problems.

B.1.3 Preconditioned Conjugate Gradient

Performance of linear CG method is highly dependent on the conditioning of the problem. As a result in practice, this method is usually used with an appropriate preconditioner which corrects the conditioning of the problem and improves the convergence significantly. Preconditioner may be applied to CG method, according to Algorithm B.1.3, (Nocedal et Wright, 1999):

Algorithm B.1.3

For a linear system of equations $\mathbf{Ax} = \mathbf{b}$
Choose an initial value $\mathbf{x}_0$, and a preconditioner $\mathbf{M}$
Calculate the residual: $\mathbf{r}_0 = \mathbf{Ax}_0 - \mathbf{b}$
Solve $\mathbf{My}_0 = \mathbf{r}_0$ for $\mathbf{y}_0$, Set $\mathbf{p}_0 = -\mathbf{y}_0$, $k = 0$;
while $\mathbf{r}_k \neq 0$

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{y}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k$$
Solve $\mathbf{My}_{k+1} = \mathbf{r}_{k+1}$
$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{y}_{k+1}}{\mathbf{r}_k^T \mathbf{y}_k}$$
$$\mathbf{p}_{k+1} = -\mathbf{y}_{k+1} + \beta_{k+1} \mathbf{p}_k$$
$$k = k + 1$$

end(while)

B.1.4 Nonlinear Conjugate Gradient

For problems with non-quadratic objective function, one can not use the linear conjugate gradient method. In such cases, Nonlinear Conjugate Gradient (NL-CG) can be used. There are different types of NL-CG introduced in the literature which their difference is basically in how they choose β_k for obtaining the new conjugate vector which specifies the direction of descent. The three best known methods for choosing β are:

- Fletcher-Reeves (FR)
- Polak-Ribière (PR)
- Hestenes-Stiefel (HS)

This method is very similar to the linear version except for changes in calculation of the step length α, and residual which is the gradient of nonlinear function in this case. Following Algorithm

shows how this method works for nonlinear optimization when Fletcher-Reeves (FR) method have been used (Nocedal et Wright, 1999):

Algorithm B.1.4

Choose an initial value $\mathbf{x}_0$
Evaluate $f_0 = f(\mathbf{x}_0), \nabla f_0 = \nabla f(\mathbf{x}_0)$
Set $\mathbf{p}_0 = -\nabla f_0$, $k = 0$;
while $\nabla f_k \neq 0$
    Compute α_k and set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

    Evaluate ∇f_{k+1};

        $\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$
        $\mathbf{p}_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{FR} \mathbf{p}_k$
        $k = k + 1$
end(while)

This algorithm is identical to that of linear conjugate gradient B.1.2, if f is chosen to be quadratic and α_k is the exact minimizer($\nabla f_k^T \mathbf{p}_{k-1} = 0$). If the line search which will be used to calculate the α_k satisfies the strong Wolf conditions (Nocedal et Wright, 1999), it is guaranteed that the all directions $\mathbf{p}_k$ are descent directions and minimize f.

Below, formulas for calculating β_{k+1} in PR and HS are given. As explained for FR, these are all identical to their linear counterpart if above conditions are met.

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}, \qquad \beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T \mathbf{p}_k} \tag{B.3}$$

Comparing these methods, PR and HS have been seen to perform similar, both theoretically and practically, however PR outperforms FR. This is because in PR, a restart is performed if a bad direction is detected. This is not the case for FR and if the initial value is not chosen properly, this method can perform poorly. PR+ is an extension of PR to guarantees the descent direction. One can refer to Chapter 5 of (Nocedal et Wright, 1999) for more discussion on different formulas proposed for calculating β_{k+1}.

B.2 Quasi-Newton Method

As discussed in 2.4.1, beside all the advantages of conjugate gradient method, this method does not guarantee the non-negativity-constraint required for medical images. Moreover, the convergence of this method is linear and for hard problems where the condition number is usually large, an appropriate preconditioner is required to preserve the convergence property of this method. One

alternative, is to use L-BFGS-B from Quasi-Newton family to benefit from their super-linear convergence. Similar to Steepest descent method, here only the gradient of the objective function needs to be calculated at every iteration. The version with bound-constraint can also be used to address the non-negativity issue. The minimizer or direction for this family of optimization is:

$$\mathbf{p}_k = -\mathbf{B}_k^{-1}\nabla f_k$$

where $\mathbf{B}_k$ is used in place of true Hessian and is a symmetric positive definite matrix. One can also use the inverse of this approximation to the true Hessian ($\mathbf{H}_k \triangleq \mathbf{B}_k{}^{-1}$) for calculations.

B.2.1 BFGS Method

In this method instead of calculating the exact hessian as in Newton's method, an approximate hessian is calculated and updated at each iteration using a low-rank formula. This method also have an effective self-correcting property and will correct the direction if the convergence becomes slow. Below is the algorithm for this method. One simple choice for Inverse Hessian approximation $\mathbf{H}_0$ is to use the identity matrix. Having a positive-definite $\mathbf{H}_k$ will guarantee that $\mathbf{H}_{k+1}$ will also be positive-definite, so choosing the initial value properly this condition will hold (for proof please see pg 141,(Nocedal et Wright, 1999)).

Algorithm B.2.1

Choose an initial value $\mathbf{x}_0$
Choose the convergence tolerance $\varepsilon > 0$
  Inverse Hessian approximation $\mathbf{H}_0$
$k = 0$;
while $\|\nabla f_k\| > \varepsilon$
  Compute search direction
    $\mathbf{p}_k = -\mathbf{H}_k \nabla f_k$;

  Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ where α_k is computed from a line search
  procedure to satisfy the Wolf condition (Nocedal et Wright, 1999)
  Define $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$
  Set $\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$
  Compute $\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T)\mathbf{H}_k(I - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T$
    $k = k + 1$
end(while)

B.2.2 L-BFGS Method

For large-size problems where calculation of Hessian matrix in a reasonable cost is not possible or the Hessian is not sparse, the limited memory BFGS can be used. In this method one does not need

to store the whole Hessian matrix but a few vectors that represent the approximation implicitly will be enough. This is expected to reduce the convergence rate of BFGS but linear convergence have been observed for large-size problems. Below the algorithm for using this method is provided (Nocedal et Wright, 1999).

Algorithm B.2.2

Choose an initial value $\mathbf{x}_0$, integer $m > 0$
$k = 0$;
repeat
Define $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$
$\rho = \frac{\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^T \mathbf{y}_{k-1}}$
$\mathbf{H}_0 = \rho \mathbf{I}$
Compute $\mathbf{p}_k = -\mathbf{H}_k \nabla f_k$;
Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ where α_k is chosen to
satisfy the Wolf condition (Nocedal et Wright, 1999)
if $k > m$
Discard th vector pair $\{\mathbf{s}_{k-m}, \mathbf{y}_{k-m}\}$ from storage
Compute and save $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$
$k = k + 1$
until convergence

Here m defines the number of most recent correction pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$ which will be used for Hessian approximation at every iteration. It is recommended to choose this number between 3-20. This method have also seen to outperform the Hessian-free Newton methods and as a result is a good choice for large sparse problems. For $m = 1$, this algorithm is equivalent to HS-Nonlinear conjugate gradient method.

APPENDIX C

PENALIZATION OF NEIGHBORING PIXELS

Assuming that image is discretized in following form:

$$Image = \underbrace{\begin{bmatrix} x_1 & x_{K+1} & \dots & x_{\frac{KN}{2}+1} & \dots & x_{K(N-1)+1} \\ x_2 & x_{K+2} & \dots & \vdots & & \vdots \\ \vdots & & & & & \\ x_K & \dots & & & & x_{KN} \end{bmatrix}}_{K \times N} \tag{C.1}$$

Matrices of the first differences, have the following forms::

$$\mathbf{D}^{(1)} = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & & \\ & & & 1 & -1 \end{bmatrix}}_{K-1 \times K} & & 0 \\ & \ddots & \\ 0 & & \underbrace{\begin{bmatrix} 1 & -1 & & \dots & \\ & 1 & -1 & \dots & \\ & & \ddots & & \\ & & & 1 & -1 \end{bmatrix}}_{K-1 \times K} \end{bmatrix}}_{(K-1)N \times KN}$$

$$\mathbf{D}^{(1)T}\mathbf{D}^{(1)} = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}}_{K\times K} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \underbrace{\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}}_{K\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(2)} = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} & & \mathbf{0} \\ & & \ddots & & \ddots & \\ \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} & & \mathbf{0} & & \underbrace{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{K\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(2)^T}\mathbf{D}^{(2)} =$$

$$\underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 2 & & & \\ & 2 & & \\ & & \ddots & \\ & & & 2 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} & & \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} \\ & \ddots & \ddots & \\ \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} & & \underbrace{\begin{bmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} 2 & & & \\ & 2 & & \\ & & \ddots & \\ & & & 2 \end{bmatrix}}_{K\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(3)} = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \\ & & & 1 & 0 \end{bmatrix}}_{K-1\times K} & \underbrace{\begin{bmatrix} 0 & -1 & & \\ & 0 & -1 & \\ & & \ddots & \ddots \\ & & & 0 & -1 \end{bmatrix}}_{K-1\times K} & & \mathbf{0} \\ & & \ddots & & \ddots \\ \underbrace{\begin{bmatrix} 0 & -1 & & \\ & 0 & -1 & \\ & & \ddots & \ddots \\ & & & 0 & -1 \end{bmatrix}}_{K-1\times K} & & \mathbf{0} & & \underbrace{\begin{bmatrix} 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \\ & & & 1 & 0 \end{bmatrix}}_{K-1\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(4)} = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 0 & -1 & & \\ & 0 & -1 & \\ & & \ddots & \ddots \\ & & & 0 & -1 \end{bmatrix}}_{K-1\times K} & \underbrace{\begin{bmatrix} 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \\ & & & 1 & 0 \end{bmatrix}}_{K-1\times K} & & \mathbf{0} \\ & & \ddots & & \ddots \\ \underbrace{\begin{bmatrix} 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \\ & & & 1 & 0 \end{bmatrix}}_{K-1\times K} & & \mathbf{0} & & \underbrace{\begin{bmatrix} 0 & -1 & & \\ & 0 & -1 & \\ & & \ddots & \ddots \\ & & & 0 & -1 \end{bmatrix}}_{K-1\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(3)^T}\mathbf{D}^{(3)} =$$

$$\underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 & & & & \\ & 2 & & & \\ & & 2 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} 0 & -1 & & & \\ & 0 & -1 & & \\ & & 0 & -1 & \\ & & & \ddots & \ddots \\ & & & & 0 \end{bmatrix}}_{K\times K} & & \underbrace{\begin{bmatrix} 0 & & & & \\ -1 & 0 & & & \\ & -1 & 0 & & \\ & & \ddots & \ddots & \\ & & & -1 & 0 \end{bmatrix}}_{K\times K} \\ & \ddots & & & \\ \underbrace{\begin{bmatrix} 0 & -1 & & & \\ & 0 & -1 & & \\ & & 0 & -1 & \\ & & & \ddots & \ddots \\ & & & & 0 \end{bmatrix}}_{K\times K} & & \underbrace{\begin{bmatrix} 0 & & & & \\ -1 & 0 & & & \\ & -1 & 0 & & \\ & & \ddots & \ddots & \\ & & & -1 & 0 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} 1 & & & & \\ & 2 & & & \\ & & 2 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}}_{K\times K} \end{bmatrix}}_{KN\times KN}$$

$$\mathbf{D}^{(4)^T}\mathbf{D}^{(4)} =$$

$$\begin{bmatrix}
\underbrace{\begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 1 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} 0 & & & & & \\ -1 & 0 & & & & \\ & -1 & 0 & & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 0 \end{bmatrix}}_{K\times K} & & 0 & & \underbrace{\begin{bmatrix} 0 & -1 & & & & \\ & 0 & -1 & & & \\ & & 0 & -1 & & \\ & & & \ddots & \ddots & \\ & & & & & -1 \\ & & & & & 0 \end{bmatrix}}_{K\times K} \\
 & & \ddots & & & \\
\underbrace{\begin{bmatrix} 0 & & & & & \\ -1 & 0 & & & & \\ & -1 & 0 & & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 0 \end{bmatrix}}_{K\times K} & & 0 & & \underbrace{\begin{bmatrix} 0 & -1 & & & & \\ & 0 & -1 & & & \\ & & 0 & -1 & & \\ & & & \ddots & \ddots & \\ & & & & & -1 \\ & & & & & 0 \end{bmatrix}}_{K\times K} & \underbrace{\begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 1 \end{bmatrix}}_{K\times K}
\end{bmatrix}}_{KN\times KN}$$

APPENDIX D

DIAGONALIZATION OF CIRCULANT MATRICES

A circulant matrix is a special kind of Toeplitz matrix and has the following form (Chen, 2005):

$$\mathbf{C}_n = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \dots & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_2 \\ c_2 & c_1 & c_0 & \dots & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \dots & c_0 \end{bmatrix} \tag{D.1}$$

Matrices of this type can be represented by first row/column and the rest of rows/columns are obtained by circ-shifting the first one. The root vector which can represent the whole matrix will then be:

$$\mathbf{c} = [c_0, c_1, \dots, c_{n-2}, c_{n-1}]^T \tag{D.2}$$

For using the circulant matrices as preconditioner, we are interested to compute $\mathbf{f} = \mathbf{C}_n\mathbf{g}$ and $\mathbf{g} = \mathbf{C}_n^{-1}\mathbf{f}$ quickly so that these operations can be used when implementing the solver with preconditioner.

Now, to show how these computations can be done quickly using FFT methods, lets define a permutation matrix $\mathbf{R}_n$ with following structure:

$$\mathbf{R}_n = \begin{bmatrix} 0 & 1 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} = \left[\begin{array}{cccc|c} 0 & 1 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{array}\right] \tag{D.3}$$

and let,

$$\mathbf{c} = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{n-1} \end{bmatrix}^T$$

Then it is easy to see that $\mathbf{R}_n = (e_2, e_3, \dots, e_n, e_1)$ is a simple permutation matrix and if multiplied by any matrix from right, it can permute the first column to last column. This means that if $\mathbf{R}_n^k = \mathbf{I}\mathbf{R}_n \dots \mathbf{R}_n$ and $\mathbf{R}_n^k$ is defined as following:

$$\mathbf{R}_n^k = \left[\begin{array}{ccc|ccc} \overbrace{} & {\scriptstyle n-k} & \overbrace{} & \overbrace{} & {\scriptstyle k} & \overbrace{} \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & 0 & \dots & 0 \end{array}\right] \tag{D.4}$$

then $\mathbf{C}_n$ can be represented as:

$$\mathbf{C}_n = \begin{bmatrix} \mathbf{c} & \mathbf{R}_n\mathbf{c} & \mathbf{R}_n^2\mathbf{c} & \dots & \mathbf{R}_n^{n-1}\mathbf{c} \end{bmatrix} \tag{D.5}$$

Using $\mathbf{R}_n^k$, we can then separate the full contribution of c_k to $\mathbf{C}_n$ to following matrices:

$$\begin{aligned} \mathbf{C}_n = &\begin{bmatrix} c_0 & & & & \\ & c_0 & & & \\ & & c_0 & & \\ & & & \ddots & \\ & & & & c_0 \end{bmatrix} + \begin{bmatrix} & & & & c_1 \\ c_1 & & & & \\ & c_1 & & & \\ & & \ddots & & \\ & & & c_1 & \end{bmatrix} + \begin{bmatrix} & & & c_2 & \\ & & & & c_2 \\ c_2 & & & & \\ & c_2 & & & \\ & & \ddots & & \\ & & & c_2 & \end{bmatrix} \\ &+ \begin{bmatrix} & & c_3 & & \\ & & & c_3 & \\ & & & & c_3 \\ c_3 & & & & \\ & c_3 & & & \\ & \ddots & & & \\ & & c_3 & & \end{bmatrix} + \ldots + \begin{bmatrix} & c_{n-1} & & & \\ & & c_{n-1} & & \\ & & & \ddots & \\ & & & & c_{n-1} \\ c_{n-1} & & & & \end{bmatrix} \\ &= c_0\mathbf{I} + c_1\mathbf{R}_n + c_2\mathbf{R}_n^2 + \ldots + c_{n-1}\mathbf{R}_n^{n-1} \\ &= \sum_{j=0}^{n-1} c_j\mathbf{R}_n^{n-1} \end{aligned} \tag{D.6}$$

Equation(D.6) shows the relationship between $\mathbf{C}_n$ and $\mathbf{R}_n$. We can also show that there is a simple relationship between $\mathbf{R}_n$ and the discrete Fourier Transform matrix (DFT), $\mathbf{F}_n$. Let denote the vector $\mathbb{I} = e = (1,1,1,\ldots,1,1)^T$ and $\mathbf{D}_n = diag(\mathbf{d}_n)$ with $\mathbf{d}_n$ being the second column of matrix

$\mathbf{F}_n$, i.e., $\mathbf{d}_n = (1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1})^T$, where $\omega_n = exp\{-2\pi i/n\}$. Then we have:

$$\begin{aligned}\mathbf{F}_n &= \begin{bmatrix}\mathbb{I} & d_n & \mathbf{D}_n d_n & \mathbf{D}_n^2 d_n & \ldots & \mathbf{D}_n^{n-3} d_n & \mathbf{D}_n^{n-2} d_n\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{D}_n^0\mathbb{I} & \mathbf{D}_n^1\mathbb{I} & \mathbf{D}_n^2\mathbb{I} & \mathbf{D}_n^3\mathbb{I} & \ldots & \mathbf{D}_n^{n-2}\mathbb{I} & \mathbf{D}_n^{n-1}\mathbb{I}\end{bmatrix}\end{aligned} \tag{D.7}$$

By Noting that $\mathbf{D}_n^n = \mathbf{D}_n^0 = diag(\mathbb{I})$, i.e., $\omega_n^n = 1$, we get the following relationship for $\mathbf{D}_n$ and $\mathbf{R}_n$:

$$\begin{aligned}\mathbf{D}_n\mathbf{F}_n &= \mathbf{D}_n\begin{bmatrix}\mathbf{D}_n^0\mathbb{I} & \mathbf{D}_n^1\mathbb{I} & \mathbf{D}_n^2\mathbb{I} & \mathbf{D}_n^3\mathbb{I} & \ldots & \mathbf{D}_n^{n-2}\mathbb{I} & \mathbf{D}_n^{n-1}\mathbb{I}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{D}_n^1\mathbb{I} & \mathbf{D}_n^2\mathbb{I} & \mathbf{D}_n^3\mathbb{I} & \mathbf{D}_n^4\mathbb{I} & \ldots & \mathbf{D}_n^{n-1}\mathbb{I} & \mathbf{D}_n^n\mathbb{I}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{D}_n^1\mathbb{I} & \mathbf{D}_n^2\mathbb{I} & \mathbf{D}_n^3\mathbb{I} & \mathbf{D}_n^4\mathbb{I} & \ldots & \mathbf{D}_n^{n-1}\mathbb{I} & \mathbf{D}_n^0\mathbb{I}\end{bmatrix} \\ &= \mathbf{F}_n\mathbf{R}_n\end{aligned} \tag{D.8}$$

which implies:

$$\mathbf{F}_n\mathbf{R}_n\mathbf{F}_n^{-1} = \mathbf{D}_n \tag{D.9}$$

This means that using the DFT matrix $\mathbf{F}_n$ one can diagonalize the permutation matrix $\mathbf{R}_n$ and so obtain the eigenvalues and eigenvectors of $\mathbf{R}_n$ easily. Note that for eigenvalues $\lambda(\mathbf{R}_n^j) = \lambda(\mathbf{R}_n)^j$ (this results can be obtained using similar way of proving the Cayley-Hamilton theorem in linear algebra), so having the eigenvalues of $\mathbf{R}_n$ we can obtain that of $\mathbf{R}_n^j$.

Now, using the result obtained in (D.9) and the relationship (D.6), one can get (Theorem 2.5.9 of (Chen, 2005)):

$$\begin{aligned}\mathbf{F}_n\mathbf{C}_n\mathbf{F}_n^{-1} &= \mathbf{F}_n\sum_{j=0}^{n-1} c_j\mathbf{R}_n^j\mathbf{F}_n^{-1} \\ &= \sum_{j=0}^{n-1} c_j(\mathbf{F}_n\mathbf{R}_n^j\mathbf{F}_n^{-1})^j = diag(\mathbf{F}_n\mathbf{c})\end{aligned} \tag{D.10}$$

In the above equation the facts that $\mathbf{F}_n\mathbf{c} = \sum_{j=0}^{n-1} c_j\mathbf{D}_n^j\mathbf{I}$ and $\mathbf{D}_n^j$ represents the diagonal of diagonal matrix $\mathbf{D}_n^j$ have been used.

To summarize, the above expressions show that eigenvalues and eigenvectors of a circulant matrix can be obtained in a fast way using DFT method. This allows fast multiplication of a circulant matrix with a vector possible:

$$\mathbf{f} = C_n\mathbf{g} = \mathbf{F}_n^{-1}diag(\mathbf{F}_n\mathbf{c})\mathbf{F}_n\mathbf{g} = \mathbf{F}_n^{-1}[(\mathbf{F}_n\mathbf{c}).*(\mathbf{F}_n\mathbf{g})] \tag{D.11}$$

Multiplication of the inverse of this circulant matrix by any vector, can then be performed as

following:

$$\mathbf{g} = \mathbf{C}_n^{-1}\mathbf{f} = \mathbf{F}_n^{-1} diag(1./(\mathbf{F}_n\mathbf{c}))\mathbf{F}_n\mathbf{f} = \mathbf{F}_n^{-1}[\mathbf{F}_n\mathbf{f}./\mathbf{F}_n\mathbf{c}] \tag{D.12}$$

Here $\mathbf{C}_n$ should be non-singular to avoid singularity problem.

APPENDIX E

CHOLESKY DECOMPOSITION

The lower triangular matrix $\mathbf{L}$ in Cholesky decomposition is calculated as following (Bertsekas, 1999): Assume that a_{ij} represent the elements of system matrix $\mathbf{A}$, and $\mathbf{A}_i$ be the i-th leading principle submatix of $\mathbf{A}$ as below:

$$\mathbf{A}_i = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} \\ a_{21} & a_{22} & \dots & a_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ii} \end{bmatrix}$$

The system matrix $\mathbf{A}$ is successively factorized to $\mathbf{A}_1, \mathbf{A}_2, \dots$ and matrix of lower-triangular $\mathbf{L}_i$ is calculated for each $\mathbf{A}_i$, i.e., $\mathbf{L}_1 = \left[\sqrt{a_{11}}\right]$ and $\mathbf{A}_1 = \mathbf{L}_1\mathbf{L}_1^T$. Then representing the Cholesky factorization of $\mathbf{A}_{i-1}$ by:

$$\mathbf{A}_{i-1} = \mathbf{L}_{i-1}\mathbf{L}_{i-1}^T \tag{E.1}$$

$\mathbf{A}_i$ can be written as:

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{A}_{i-1} & \boldsymbol{\beta}_i \\ \boldsymbol{\beta}_i^T & a_{ii} \end{bmatrix}, \quad \boldsymbol{\beta}_i = \begin{bmatrix} a_{1i} \\ \vdots \\ a_{i-1,i} \end{bmatrix} \tag{E.2}$$

From Eq.(E.1) and (E.2) we can then write:

$$\mathbf{A}_i = \mathbf{L}_i\mathbf{L}_i^T \tag{E.3}$$

where,

$$\mathbf{L}_i = \begin{bmatrix} \mathbf{L}_{i-1} & 0 \\ l_i^T & \lambda_{ii} \end{bmatrix} \tag{E.4}$$

and,

$$l_i = L_{i-1}^{-1}\beta_{ii}, \quad \lambda_{ii} = \sqrt{a_{ii} - l_i^T l_i}$$

λ_{ii} is well defined as $a_{ii} - l_i^T l_i$ is always positive (for proof please see (Bertsekas, 1999), Appendix D). Note that for lower-triangular matrices, their inverse can be analytically found, so $\mathbf{L}_{i-1}^{-1}$ is easily invertible.

Remark 5.3.2: Note that for positive-definite $\mathbf{A}$, the submatrix $\mathbf{A}_y$ is also positive-definite as

for any $\mathbf{y} \in \mathbb{R}_i$ and $\mathbf{y} \neq 0$ we have:

$$\mathbf{y}^T \mathbf{A}_y \mathbf{y} = \begin{bmatrix} \mathbf{y}^T 0 \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} > 0$$

This shows that calculation of lower-triangular in this method is unique among all its similar counterparts and it is numerically efficient.