ACCELERATED STATISTICAL IMAGE RECONSTRUCTION ALGORITHMS AND SIMPLIFIED COST FUNCTIONS FOR X-RAY COMPUTED TOMOGRAPHY

by

Somesh Srivastava

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2008

To my parents and sister.

ACKNOWLEDGEMENTS

This thesis would not be possible without the direct and indirect help and support of a lot of people. First and foremost is Prof. Jeff Fessler, whose invaluable guidance and constant support as a research advisor helped me take the first steps in engineering research. Jean-Baptiste Thibault, Ph.D., (CT Scientist, GE Healthcare) collaborated closely with us on one of the investigatations and provided practical knowledge about actual X-ray CT scanners, scanner data, and image reconstruction software libraries. I also wish to acknowledge the insightful discussions I had with Roy Nilsen (GE Healthcare), Bruno De Man, Ph.D., (GE Global Research), and with members of Prof. Randy Ten Haken's research group at the Radiation Oncology Dept., University of Michigan, Ann Arbor. Professors at the University of Michigan, Ann Arbor, serving in various capacities, helped me understand key concepts and research various topics. I am grateful to the current and former students of Prof. Fessler's research group for many enlightening presentations and discussions. Staff members of the Electrical Engineering and Computer Science (EECS) Department and University of Michigan provided prompt support that helped the day-to-day research activity proceed smoothly. Special thanks are due to Don Winsor, Ph.D., and Elizabeth Zaenger of the Departmental Computing Organization (DCO) for maintaining the computers used in this research. I am grateful to my parents and my sister for the constant encouragement and moral support without which the work for this thesis would not be possible.

TABLE OF CONTENTS

DEDICATION . . . . . **ii**

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF FIGURES . . . . . **vii**

LIST OF TABLES . . . . . **x**

ABSTRACT . . . . . **xi**

CHAPTER

1. Introduction 1

1.1 Contributions 6

2. Background 9

2.1 Basic concepts 9
2.2 Sampling 13
2.3 Assigning a statistical model to the observations 15
2.4 Choosing a system model 16
2.5 Including physical effects like beam hardening 17
2.6 Formulating a cost function 19
2.6.1 Regularization 20
2.6.2 Cost function properties 22
2.7 Choosing a minimization algorithm 23
2.7.1 Iterative minimization using optimization transfer 25
2.7.2 Ordered subsets (OS) 26
2.7.3 Preconditioned conjugate gradient (PCG) 27
2.7.4 Iterative Coordinate Descent (ICD) 29
2.7.5 Non-negatively constrained algorithms 30
2.7.6 Summary of algorithm properties 31
2.8 Cone-beam geometry 31

3. Cost function for the non-negativity constraint 34

3.1 Increase of compute time in PCG algorithms due to the non-negativity constraint . . 35
3.2 Cost function modification 36
3.3 Simulations and results 39
3.4 Summary 41

4. Simplified cost function for beam hardening 44

4.1 Sinogram pre-correction for beam hardening : Joseph-Spital (JS) method 44
4.2 Statistical reconstruction methods 47
4.2.1 1-parameter and Ad hoc methods 48
4.2.2 2-parameter and Exact methods 49
4.3 Simulation and results 50
4.4 Summary 51

5. Statistical method to reduce organ motion artifacts 55

5.1 Background 55
5.2 Problem definition 57
5.3 Proposed statistical reconstruction method 58
5.4 Results 61
5.5 Summary 62

6. Comparisons between OS and PCG algorithms 69

6.1 Conjugate-gradient algorithms 69
6.1.1 PCG_LS_QS (or PLQ) 69
6.1.2 PWLS_PCG_LS_QS (or PWLS_PLQ) 71
6.1.3 QS_PCG_LS (or QPL) 72
6.1.4 Preconditioners 75
6.1.5 Comparison between PCG_LS_QS and QS_PCG_LS 76
6.2 Ordered-subset algorithms 76
6.2.1 OS_SPS algorithm for an unregularized cost function 76
6.2.2 PWLS_OS_SPS algorithm 79
6.3 Comparison between PWLS_OS_SPS and PWLS_PLQ 81
6.4 Summary 83

7. Algorithm acceleration using hybrid-algorithm and parallel-computation approaches . 86

7.1 Hybrid-algorithm approach 87
7.1.1 Basic principle 87
7.1.2 OS-ICD hybrid algorithm 87
7.2 Parallel-computation approach 88
7.2.1 Parallel PWLS_OS_SPS 89
7.2.2 Implementation aspects of the Parallel PWLS_OS_SPS algorithm 94
7.3 Results 99
7.3.1 Scanner and dataset description 99
7.3.2 Hardware and software 100
7.3.3 Reconstructed images 100
7.3.4 Compute time of the algorithms 102
7.3.5 Candidate methods to accelerate the OS-algorithm 104
7.3.6 Reconstructions from OS-based algorithms in the end slices 106
7.4 Summary 107
7.5 Future work 108
7.5.1 Modification of OS algorithms for helical cone-beam geometry 108
7.5.2 Modification of distance-driven projector for oblique rays 109

8. Algorithm designs for even more acceleration 111

8.1 OS-PCG hybrid algorithm: OS_PCG_LS 111
8.2 PWLS_OS_SPS with reduced number of non-linear function evaluations 114
8.3 Surrogate cost function for reduced inter-processor communication 124

8.3.1 Application to targeted reconstruction 132
8.3.2 Overheads 133

9. Conclusions and future work 135

9.1 Conclusions 135
9.2 Future work 137

APPENDIX **140**

BIBLIOGRAPHY **170**

LIST OF FIGURES

Figure

2.1 Schematic diagram of a X-ray scanner. 10

2.2 Axial and helical cone beam X-ray CT. 32

3.1 Plots of $h_i(t)$ (solid line) and $\tilde{h}_i(t)$ (dashed line) for the three categories. $r_i = 10$, $b_i = 8$, and $y_i = 4, 15$ and 25 in (a), (b) and (c) above respectively. Note that $\tilde{h}_i = h_i$ for $t \geq 0$ and differs only for non-physical values of the attenuation coefficients. 36

3.2 Images of the "True Phantom" and its reconstructions by various algorithms. 41

3.3 The difference images between the reconstructions and the true phantom. *NRMSE* is defined as $\|[\boldsymbol{\mu}^{(n)}]_+ - \boldsymbol{\mu}_{true}\| / \|\boldsymbol{\mu}_{true}\|$. 42

3.4 Plot showing the variation of $\Phi(\boldsymbol{\mu})$ with iteration number and CPU time. 42

3.5 Plot showing the variation of $\Phi([\boldsymbol{\mu}]_+)$ with iteration number and CPU time. 43

3.6 Plot showing the variation of number of negative pixels and the average value of negative pixels with iteration number. 43

4.1 Plot of $f(T_S, T_B)$ and $\gamma(T_S, T_B)$ 47

4.2 Image of the true phantom and JS water-corrected FBP reconstruction. Window = 400HU. Note the beam-hardening streak in the center of the FBP reconstruction. 52

4.3 Reconstruction using statistical methods. Window = 400HU. *Clockwise from top left :* 1-parameter, 2-parameter, Ad hoc, Exact. 53

4.4 Difference images between reconstructions using statistical methods and the true phantom. *Clockwise from top left :* 1-parameter, 2-parameter, Ad hoc, Exact. 54

5.1 True images. *Left:* Object in state 1, and *Right:* Object in state 2. Image window = $[800, 1200]$ HU. 63

5.2 Motion mask computation procedure. *Top-left:* Thresholded residual sinogram, *Top-right:* Back projection of thresholded residual sinogram, and *Bottom-left:* Motion mask. 64

5.3 Computing the transition view. *Left:* Residual sinogram for rays passing through motion mask, and *Right:* Sum of sinogram on the left over the detector dimension. 65

5.4 FBP reconstruction and motion mask. *Left:* FBP reconstruction, and *Right:* FBP reconstruction with motion mask superimposed. Image window = $[800, 1200]$ HU. 66

5.5 Iterative reconstruction. *Left:* Motion state 1, and *Right:* Motion state 2. Image window = $[800, 1200]$ HU. 67

5.6 Error images. *Top-left:* True motion state 1 - iterative motion state 1, *Top-right:* True motion state 2 - iterative motion state 2, *Bottom-left:* True motion state 1 - FBP, and *Bottom-right:* True motion state 2 - FBP. Image window = $[-30, 30]$ HU. 68

6.1 Convergence rate comparison between PCG_LS_QS and QS_PCG_LS algorithms 76

6.2 Reconstructions from statistical algorithms. Window = 400HU. 83

6.3 Difference plots between statistical reconstructions and final reconstruction, 1000 iterations. *Left*: L_1^{avg}, *Right*: L_∞. 84

6.4 Difference plots between statistical reconstructions and final reconstruction, first 100 iterations. *Left*: L_1^{avg}, *Right*: L_∞. 85

7.1 Division of image and sinogram spaces among $P = 4$ processors. Longitudinal *i.e.*, z-axis, view is presented here. 90

7.2 Illustration of $\boldsymbol{\mu}_{p,p'}$ for $P = 4$ processors. 91

7.3 Image and observation space division for $P = 4$ MPUs 96

7.4 The `sync` operation. 98

7.5 The `sum` operation. 99

7.6 FBP image and intermediate image from hybrid OS-ICD algorithm. *Top:* Initial FBP image, and *Bottom:* Intermediate reconstruction. Image window = $[800, 1200]$ HU. 102

7.7 Hybrid OS-ICD and ICD reconstructions. *Top:* OS-ICD reconstruction, and *Bottom:* ICD reconstruction. Image window = $[800, 1200]$ HU. 103

7.8 A coronal slice of the image reconstruction using the OS 256-subset algorithm. 107

A.1 Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for $i \in I_1$. 144

A.2 Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_2. 145

A.3 Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_3 and maximum of $\dot{h}_i$ occurs in the positive quadrant. . 147

A.4 Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_3 and maximum of $\dot{h}_i$ occurs in the negative quadrant. 147

A.5 Representative illustration for Case 1.1($i \in I_1, s \leq 0$). 151

A.6 Representative illustration for Case 1.2($i \in I_1, s > 0$). 152

A.7 Representative illustration for Case 2.1($i \in I_2, s \leq 0$). 155

A.8 Representative illustration for Case 2.2($i \in I_2, 0 < s \leq t^*$). 157

A.9 Representative illustration for Case 2.3($i \in I_2, t^* < s$). 159

A.10 Representative illustrations for Case 3.1($i \in I'_3, s \leq 0$), Case 3.2($i \in I'_3, 0 < s \leq t^*$) and Case 3.3($i \in I'_3, t^* < s$). 161

A.11 Representative illustrations for Case 4.1($i \in I''_3, s \leq 0$) and Case 4.2($i \in I''_3, 0 < s$). . . . 167

LIST OF TABLES

Table

2.1 Expressions for potential functions, $\psi(t)$, and corresponding weighting functions, $\omega_\psi(t)$. . . 21

2.2 Properties of candidate algorithms for X-ray CT. 31

6.1 Number of iterations required to satisfy convergence criteria 82

7.1 Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 1 subset and 4 different regularization functions. 105

7.2 Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 41 subsets and 4 different regularization functions. 106

7.3 Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 256 subsets and 4 different regularization functions. 106

8.1 Skeleton of algorithm for reducing non-linear function evaluations 120

8.2 Algorithm for reducing non-linear function evaluations 123

8.3 Skeleton of parallel algorithm for reducing inter-processor communication 132

ABSTRACT

ACCELERATED STATISTICAL IMAGE RECONSTRUCTION ALGORITHMS AND SIMPLIFIED COST FUNCTIONS FOR X-RAY COMPUTED TOMOGRAPHY

by

Somesh Srivastava

Chair: Prof. Jeffrey A. Fessler

Statistical image reconstruction methods are poised to replace traditional methods like filtered back-projection (FBP) in commercial X-ray computed tomography (CT) scanners. Statistical methods offer many advantages over FBP, including incorporating physical effects and physical constraints, modeling of complex imaging geometries, and imaging at lower X-ray doses. But, the use of statistical methods is limited due to many practical problems. This thesis proposes methods to improve four aspects of statistical methods: reconstruction time, beam hardening, non-negativity constraints, and organ motion. To reduce the reconstruction time, several novel iterative algorithms are proposed that are adapted to multi-core computing, including a hybrid ordered subsets (OS) / iterative coordinate descent (ICD) approach. This approach leads to a reduction in reconstruction time, and it also makes the ICD algorithm robust to the initial guess image. Statistical methods have accounted for beam hardening by using more information than needed by traditional FBP-based methods like the Joseph-Spital (JS) method. This thesis proposes a statistical method that uses exactly the same beam hardening information as the JS method while suppressing beam hardening artifacts. Directly imposing the non-negativity constraints can increase the computation time of algorithms such as the preconditioned conjugate gradient (PCG) method. This thesis proposes a modification of the penalized-likelihood cost function for monoenergetic transmission tomography, and a corresponding PCG algorithm, that reduce reconstruction time when enforcing nonnegativity. Organ motion during a scan causes image artifacts, and in some cases these artifacts are more apparent when standard statistical methods are used. A preliminary simulation study of a new approach to remove motion artifacts is presented. The distinguishing feature of this approach is that it does not require any new information from the scanner. The target applications of this research effort are 3-D volume reconstructions for axial cone-beam and helical cone-beam scanning geometries of multislice CT (MSCT) scanners.

CHAPTER 1

Introduction

X-ray Computed Tomography (CT) is a valuable tool in medical diagnosis. It produces sharp millimeter scale resolution images of the anatomy of a patient by imaging a physical property of the body tissues known as the X-ray attenuation coefficient. Recent advancements in X-ray imaging technology have produced spectacular medical applications. Foe example, videos of the heart have been produced using Multislice CT (MSCT) scanners, due to the development of high-speed gantries and multi-row detectors. All these technological developments and applications pose new challenges to image reconstruction.

Image reconstruction is the process of mathematically computing the image of the patient or object being scanned in a X-ray CT scanner. The inputs to the mathematical computation are the observations produced by the scanner, its physical characteristics, and its instrument settings for a particular scan. Image reconstruction faces many challenges from the latest scanners. One challenge is the complex nature of the data collection geometries. Another one arises from the need to reduce the amount of X-ray exposure to the patient, also known as the X-ray dose. A reduced X-ray dose is considered better for the general well-being of the patient. Also, competing imaging technologies like Magnetic resonance imaging (MRI) do not use ionizing radiation and are almost completely harmless. Unless image reconstruction methods are improved, reconstructed images at reduced X-ray dose are noisy. This thesis explores new image reconstruction methods that try to overcome the above challenges and many more.

Two major families of image reconstruction methods for X-ray CT are the Filtered back-projection (FBP) methods, and the Statistical methods. FBP is the traditional method of image reconstruction, whereas statistical methods are more recent. Statistical methods are much more flexible than FBP, because modeling observation statistics, data collection geometry, physical effects, and physical constraints on the image is easier with statistical methods. Derivation of FBP-based methods that model all of the above has proved to be difficult. But, statistical methods suffer from one major drawback when compared to FBP–excessive computation time. A major portion of the research effort presented in this thesis is aimed at reducing the computation time of statistical methods.

A statistical image reconstruction method consists of two components: a cost function and an algorithm. The cost function is a mathematical function that maps the reconstructed image into the real-number line. The reconstructed image is a general term that can refer to a slice through the patient, or a stack of slices (also known as a volume), or a time sequence of a slice or a volume. The observations made during a scan, physical characteristics of the scanner, and scanner instrument settings participate as components of the cost function. Cost functions in X-ray CT are considered to have the property that their minimum/minima are close to the true nature of patient anatomy. The algorithm is a numerical method that finds the image that minimizes the cost function. That is how an algorithm together with a cost function produces an image of the patient. Algorithms that minimize the cost function in one step are rare. Most algorithms are iterative, *i.e.*, they start with a coarse initial guess and refine it over and over in such a way that the refinements possess lower and lower values of the cost function. The computation that produces each consecutive refinement is called an iteration. An iterative algorithm executes iterations until certain criteria, known as convergence criteria, are met. This thesis investigates new cost functions and iterative algorithms for X-ray CT.

The computation time of an iterative algorithm is a product of two numbers: the number of iterations and the average time taken to execute one iteration. The number of iterations required by an iterative algorithm to meet a certain convergence criteria roughly defines its convergence rate. Thus, using iterative algorithms with faster convergence rates can reduce the overall compute time. The following algorithms are known to have high convergence rate apart from other desirable properties: Ordered-subsets (OS, also known as block-gradient), Preconditoned conjugate gradient (PCG), and Iterative coordinate descent (ICD). Convergence rate can be further increased by creating new algorithms that somehow combine one or more of the algorithms listed above. This approach of combining algorithms, called the hybrid-algorithm approach here, is developed and tested in this thesis.

But, improvements in convergence rate alone are not expected to be sufficient to make iterative algorithms practical. The time taken to execute one iteration must also be reduced. Compute time per iteration can be reduced by using faster computers. And faster computers nowadays, are parallel, *i.e.*, they are constructed by creating a network of a large number of microprocessors. This approach of using a parallel computer to reduce per-iteration compute time is termed as the parallel-computation approach here. To summarize, this thesis explores two approaches to reduce the compute time of a statistical image reconstruction method: the hybrid-algorithm approach and the parallel-computation approach.

While reducing computation time is necessary to make statistical methods practical, it is also necessary to adapt them to correct for various kinds of image artifacts. An image artifact can be broadly defined as the systematic deviation of the reconstructed image from the true nature of the patient or object being scanned. Image artifacts can possibly obscure and/or distort important image features, which could in turn effect the patient diagnosis. Artifacts appear in images when the reconstruction method, whether FBP or Statistical, fails to account for one or more physical properties and/or physical effects in its mathematical computations. Out of the many physical properties and physical effects that when ignored can cause artifacts, three are studied in this thesis: non-negative nature of the X-ray attenuation coefficient, beam hardening, and organ motion.

X-ray attenuation coefficient is the physical property whose image is created by an X-ray CT scanner. It has non-negative values. If an image reconstruction method does not impose the non-negativity constraint on the image voxels (or, pixels), some voxels, especially those in the air regions, can become negative. This is a non-physical result and hence may be undesirable. Once can impose the non-negativity constraint on the image by using non-negatively constrained cost-function minimization methods. The imposition of the non-negativity constraint is trivial in OS-based and ICD-based algorithms. But, the direct imposition of the non-negativity constraint in PCG-based algorithms causes the compute time to increase by nearly 50%. In this thesis, a simple modification of the cost function is developed. Applying PCG-based algorithms to this modified cost function controls negative pixels to some extent and does not incur a large compute time penalty.

Beam hardening is accounted for in FBP using a post-processing method called the Joseph and Spital (JS) method. The JS method can be applied to statistical methods provided beam hardening information is excluded from the cost function. A better strategy is to include beam hardening in the cost function rather than work around beam hardening using the JS method. Inclusion of beam hardening in the cost function is especially beneficial when the X-ray dose is low. A cost function that includes beam hardening is developed and studied in this thesis.

Organ motion can cause artifacts in images produced using cost functions that ignore its occurrence. In chest scans, lungs and the heart are almost always moving while the scan is taken. One solution would be to stop or reduce organ motion using external instruments like those used for breath-hold techniques. But, not all scans can be taken while using such techniques. An alternative solution would be to detect organ motion from the data collected by the scanner and model it within the cost function. One such solution is developed and tested for a simulated, single-slice, fan-beam scan in this thesis.

X-ray CT scanners targeted by the statistical image reconstruction methods developed in this thesis are the Multislice CT (MSCT) (also known as, Multidetector row CT (MDCT)) X-ray scanners. These scanners are the current state-of-the-art in gantry based X-ray scanners. Diagnostic applications of these scanners like cardiovascular require non-standard data-collection geometries. Statistical methods handle non-standard geometries better than FBP, making them indispensable for image reconstruction in MSCT scanners. The statistical image reconstruction methods developed in this thesis are tested for data obtained from two standard geometries: axial cone-beam and helical cone-beam. These methods are flexible enough to handle non-standard geometries.

This thesis is organized as follows. Chapter 2 presents background information on X-ray CT and statistical image reconstruction methods. Chapter 3 presents the modified cost function developed here that controls negative pixels while maintaining a low computation time in PCG algorithms. Chapter 4 presents the investigation of the cost function that employs beam-hardening parameters used by the JS method. Chapter 5 presents the preliminary investigation of a statistical image reconstruction method to control organ-motion artifacts. Chapter 6 compares OS and PCG algorithms in simulated single-slice fan-beam scans. Chapter 7 describes the algorithm acceleration techniques developed using the hybrid-algorithm and parallel-computation approaches. Chapter 8 presents algorithm designs that can be used to further accelerate algorithms. Finally, Chapter 9 presents the conclusions of this thesis and future research work that can be done based on it.

1.1 Contributions

Chapter 3

- A new cost function and a correspoding PCG algorithm were proposed to control negative pixels in PCG algorithms. Compared to the current method of imposing the non-negativity constraint in PCG algorithms, the proposed PCG algorithm reduces execution time by a third while controlling negative pixels.

Chapter 4

- A new cost function that incorporates the beam hardening information used by the JS method was developed. Current statistical methods for beam hardening correction require more information than the JS method, whereas the proposed method uses only the information used by the JS method to correct the beam hardening artifacts.

Chapter 5

- A new statistical method to reduce organ motion artifacts was developed. Current methods for compensating for organ motion require measurement of signals other than the singoram from the scanner. Preliminary investigations of the proposed method indicate that it can compensate for organ motion by using the sinogram only.

Chapter 6

- A new PCG algorithm that is guaranteed to minimize non-quadratic cost functions was developed. Current PCG algorithms are not guaranteed to find the minimum of non-quadratic cost functions.

- A comparison of OS-based and PCG-based algorithms was carried out. It was found that OS algorithms converge faster then PCG algorithms to the solution in the initial iterations. But, the iterates of the OS algorithm stop approaching the solution after a few iterations. Thus, OS algorithms are desirable only if a quick, sub-optimal solution is sufficient.

Chapter 7

- A new algorithm, the hybrid OS-ICD algorithm, was developed, and a comparison of its properties with those of an existing algorithm, the ICD algorithm, carried out. The hybrid OS-ICD algorithm can potentially better the reconstruction time of the ICD algorithm. The hybrid OS-ICD algorithm also makes the ICD algorithm robust to the initial guess image.
- Parallel computation was demonstrated to reduce the per-iteration compute time by a large amount in OS-based algorithms. For a computer with 8 parallel processors, a speedup of around 7 was observed.

Chapter 8

- A new algorithm that combines ideas from OS and PCG was derived. This derivation is a step towards further speeding up OS algorithms.
- A new algorithm that is similar to the PWLS_OS_SPS algorithm but saves compute time by calling computationally expensive non-quadratic functions far lesser number of times was derived. This derivation is a step towards speeding up the PWLS_OS_SPS algorithm.
- A new surrogate of the cost function that separates a regularized statistical estimation problem into P regularized statistical estimation problems was derived in the context of parallel computation for X-ray CT. The number of times the P problems exchange values of iterates of the parameter segments they are individually responsible for

can be controlled with this new surrogate. Thus, the reduction of inter-processor communication in parallel computers can be investigated using the new surrogate.

CHAPTER 2

Background

This chapter describes the basic working of an X-ray CT imaging system and the statistical method of image reconstruction. Section 2.1 describes the basic concepts behind an X-ray CT system. Section 2.2 describes the issues surrounding the sampling of the observation space and the parameter space (*i.e.*, the space of the reconstructed quantity). The statistical reconstruction method computes the reconstructed image by the following steps.

1. Assigning a statistical model to the observations (Section 2.3)
2. Choosing a system model (Section 2.4)
3. Including physical effects like beam hardening (Section 2.5)
4. Formulating a cost function (Section 2.6)
5. Choosing a minimization algorithm (Section 2.7)

Section 2.8 describes the reconstruction issues and analytical reconstruction methods in cone-beam geometry.

2.1 Basic concepts

X-ray CT produces images of the X-ray attenuation coefficient of the object or patient being scanned. A typical construction of a X-ray scanner involves a source and a detector

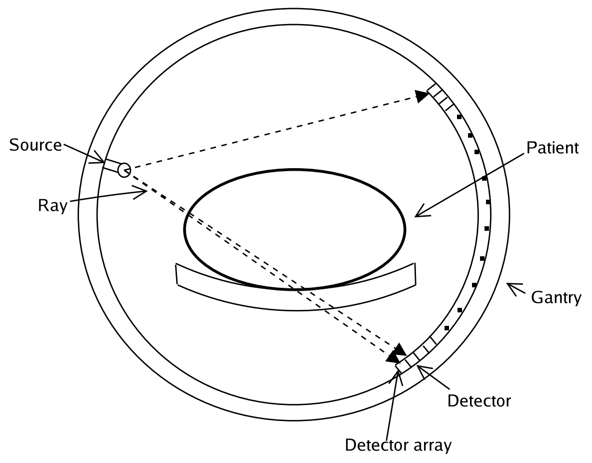


Figure 2.1: Schematic diagram of a X-ray scanner.

array (see Fig. 2.1). The source and the detector array are fixed with respect to each other in space on a C-arm or a gantry and trace a path or orbit around the patient. The source is an incoherent source of X-ray radiation and detectors record the intensity of the radiation exiting the patient. As the source and the detector array scan the patient, each source position and detector element pair cause a thin beam of radiation to pass through the patient and represent one observation. Let n_d be the total number of such source position and detector element pairs and i be the index numbering them. If the intensity of this beam of radiation, before $(I_{in,i})$ and after $(I_{out,i})$ passing through the patient is known, then Beer's law provides the total attenuation experienced by the beam:

$$I_{out,i} = I_{in,i} \exp\left(- \int_{L_i} \mu(\boldsymbol{r}) d\boldsymbol{r} \right), \tag{2.1}$$

where, L_i is the path of the ray through the patient and $\mu(\boldsymbol{r})$ is the distribution of X-ray attenuation coefficient in the patient as a function of position in the co-ordinate system fixed to the room. Thus, the observations indirectly measure the line integral of the X-ray attenuation coefficient through the patient. All image reconstruction methods whether analytical (like FBP) or statistical, attempt to recover $\mu(\boldsymbol{r})$ from the observations $\{I_{out,i} : i = 1, \ldots, n_d\}$. Note that (2.1) assumes that the X-ray photons in an X-ray beam have the same energy, *i.e.*, the X-rays used are monoenergetic. This law has to be modified for the more practical case of polyenergetic X-rays as shown in Section 2.5.

The physical units of $\mu(\boldsymbol{r})$ are cm^{-1}. The more commonly used units for $\mu(\boldsymbol{r})$ are the Hounsfield units (HU). The conversion from cm^{-1} to HU is :

$$\mu(\text{in HU}) = 1000\frac{\mu - \mu_{\text{water}}}{\mu_{\text{water}} - \mu_{\text{air}}}.$$

This scale gives a value of -1000 and 0 to air and water respectively. This makes the process of representing attenuation using unsigned numbers a little cumbersome. So, we add 1000 to μ values in HU and continue to call it HU which effectively makes the formula : $\mu(\text{in HU}) = 1000(\mu - \mu_{\text{air}})/(\mu_{\text{water}} - \mu_{\text{air}})$. [1]

The sinogram of a single slice of the object is a two dimensional function that contains the line-integral values through the object. Its first argument is the source position (as an angle) and the second argument is the angular location of the detector from the center of the detector array. Note that the ideal sinogram is a function of continuous parameters whereas the observed sinogram discretizes the parameters. Thus, a finite number of integrals through the slice are being used to reconstruct samples of a continuous function $\mu(\boldsymbol{r})$. This observation intuitively indicates that the amount of detail of an object that can be reconstructed is determined by the number of sinogram bins in the observed sinogram. The linear transform that maps the continuous image to the continuous sinogram is called the Radon transform. The inversion of the Radon transform can be carried out exactly using mathematical analysis by the Central Slice Theorem or the FBP method [46]. But, discretization of the arguments in the observed sinogram leads to approximations being made to the original theory.

[1]Images stored in HU are required to be converted to cm^{-1} in order to be used in computations. Assume $\mu_{\text{air}} \approx 0cm^{-1}$ and $\mu_{\text{water}} \approx 0.2cm^{-1}$ (at 100keV) [46]. $\Rightarrow \mu = (\mu_{HU}/1000) * \mu_{water} = (\mu_{HU}/1000) * 0.2cm^{-1}$. Thus, $\mu(in\ cm^{-1}) = 2 \times 10^{-4}\mu_{HU}$.

Various deviations from the idealized model of (2.1) occur due to low dose, instrumentation features and X-ray photon scattering. Lower dose is one of the major challenges to X-ray CT. To achieve a lower dose, one or more of the following are required: (a) lower X-ray tube voltage, (b) lower X-ray tube current, and (c) lower scan time. Lower tube voltage leads to lower photon energy and increases beam-hardening. Increased beam-hardening changes (2.1) entirely (see Section 2.5). Lower tube current leads to lower signal-to-noise ratio (SNR) in the observations making the noise in the observations significant. Some kind of denoising of the observations becomes necessary [43]. An intuitively more satisfying approach, the statistical image reconstruction approach, is to estimate the parameters of the distribution of the observations, which are nothing but the image pixels, rather than denoise the observations. The approach of denoising the observations is called the sinogram precorrection approach in which the sinogram is modified prior to reconstruction in order to account for noise and/or physical effects. Both approaches have their advantages and disadvantages and are being currently investigated by various researchers. Lower scan time increases the *afterglow effect* and requires a high speed gantry and detectors with faster response times. Lower scan times also mitigate motion artifacts due to voluntary and involuntary patient motion. Instrumentation of an actual X-ray CT scanner also causes distortions in the measurements, *e.g.*, off-focal radiation, cross-talk, finite source spot size, finite detector size *etc.* [31]. Scattering of X-ray photons is also becoming significant due to the increased cone-angle of the radiation in 3-D systems [69]. In conclusion, an ideal reconstruction method should account for all of the above practical considerations to obtain an accurate reconstruction of the object or patient being imaged.

2.2 Sampling

Sampling is the process of representing a continuous image $\mu(\boldsymbol{r})$ with a finite number of bits in a computer. The continuous image $\mu(\boldsymbol{r})$ is discretized in both space (and time) and amplitude (*i.e.*, a fixed number of bits are assigned to each basis coefficient). The continuous image space is assumed to have a countable basis out of which a finite set is chosen to approximate it: $\{b_j(\boldsymbol{r}) : j = 1, \ldots, n_p\}$. $\boldsymbol{\mu}$ is the vector of coefficients of this truncated basis representation. Thus, the image is represented as:

$$\mu(\boldsymbol{r}) \approx \sum_{j=1}^{n_p} \mu_j b_j(\boldsymbol{r}).$$

The individual basis functions are located at different points in space (and time) and can be at different scales or the same scale. This choice can influence convergence rate in statistical methods if the initial guess is far away from the minimum, *e.g.*, when starting with a zero image. If the initial image is close to the minimum then basis at the finest scale suffices. Substituting the above expression for $\mu(\boldsymbol{r})$ in the line integral expression of (2.1) we have,

$$\int_{L_i} \mu(\boldsymbol{r}) d\boldsymbol{r} \approx \int_{L_i} \sum_{j=1}^{n_p} \mu_j b_j(\boldsymbol{r}) d\boldsymbol{r} = \sum_{j=1}^{n_p} \mu_j \int_{L_i} b_j(\boldsymbol{r}) d\boldsymbol{r} = \sum_{j=1}^{n_p} \boldsymbol{G}_{ij} \mu_j \triangleq [\boldsymbol{G}\boldsymbol{\mu}]_i,$$

$$\text{where, } \boldsymbol{G}_{ij} \triangleq \int_{L_i} b_j(\boldsymbol{r}) d\boldsymbol{r}.$$

Thus, the matrix $\boldsymbol{G}$ plays the role of the integration operation in (2.1) and is called the *X-ray CT system model* or *system model* for short. The operation $\boldsymbol{G}\boldsymbol{\mu}$ is known as the forward projection operation. The transpose operation of $\boldsymbol{G}$ is called the back projection operation. Forward and back projections take up most of the computation time in statistical image reconstruction methods.

The choice of the basis functions for $\mu(\boldsymbol{r})$ is important because it determines the accuracy of representation of the continuous image and also influences the implementation of the system model. One would prefer a basis representation that performs fast forward and back projections in the system model without compromising representation accuracy. The pixel basis that involves *rect* functions is one of the simplest. It is intuitively easy to understand and its consequences are thus easy to control. Its implementation in system models (see Section 2.4) is also simpler and relatively computationally inexpensive. However, its main disadvantage is that it is not smooth and its Fourier transform has very a large side-lobe amplitude. This large side-lobe amplitude could amplify errors. This is because the image reconstruction process is basically the Radon inversion operation. The inversion process carries out differentiation of sinogram data from a single row (*i.e.* angle). The magnitude of the Fourier transform of the differentiation operation in 1-dimension increases linearly with frequency with slope 1. If the object has inaccuracies in representation at higher frequencies, *e.g.* due to large side-lobes of the basis functions, these will be amplified by the differentiation operation. An alternative is to use smoother basis functions like B-splines [29]. The Radon transform for images represented on a grid (as opposed to continuous functions), as is done above, has been an object of study due to the inexact implementation of the Central slice theorem using discretized image and sinograms. A notion of Radon transform for discrete data which is both theoretically satisfactory and practically useful (very low condition numbers) is presented in [4].

Field of view (FOV) considerations are important from a practical and as well as a theoretical point of view. Here, FOV is defined as the region in space within which a given X-ray CT imaging system can reconstruct the object to a high degree of accuracy. In 2-dimensional fan-beam tomography, there exists a circle within which each pixel of the object has a ray passing through it from each position of the source. We use this circle as a rough measure of FOV. The relation between detector size, number of source angles, image resolution and FOV for fan-beam tomography was derived in [48] . In the

3-dimensional case, Tuy's condition determines the FOV (see Section 2.8).

2.3 Assigning a statistical model to the observations

The noise present in the observations can be modeled using a probability density function (or probability mass function). The observations may also be effected by other physical processes which are assumed to be accounted for using appropriate sinogram precorrection techniques. The observations y_i obtained by the scanner are pre-corrected to obtain $\check{y}_i$, which are then used for reconstruction (*e.g.* see Section 4.1). The statistics mainly depend on the physical processes operating within the detector system. A detailed analysis is presented in [19]. The models investigated in [19] are : (a) Compound Poisson and (b) Compound Poisson with Gaussian readout noise. However, the following simple models have been found satisfactory in current practice :

1. *Poisson* [1, 23] : A detailed observation model for an X-ray CT system has been investigated in [43]. We use a simplified version of [43, Eq. 9] here. For a source-detector-pair i, b_i is the initial intensity of the beam expressed as number of counts (also called blank-scan counts) and r_i accounts for the read-out noise (also called random counts) and scatter. The observations are assumed to be distributed as :

$$\check{y}_i \sim \mathsf{Poisson}\left\{b_i e^{-[\boldsymbol{G\mu}]_i} + r_i\right\} . \tag{2.2}$$

2. Weighted least squares (WLS) ([23, 56]) : The negative log-likelihood of (2.2) is approximated using a quadratic function in [23]. This quadratic approximation leads us to the following variable transformation: $\check{l}_i \overset{\triangle}{=} ln(\frac{b_i - r_i}{\check{y}_i})$. The quadratic approximation implies that $\check{l}_i$ is a Gaussian random variable. The parameters of the quadratic approximation suggest that the mean and variance of $\check{l}_i$ are $[\boldsymbol{G\mu}]_i$ and $\check{y}_i^{obs}/(\check{y}_i^{obs} - r_i)^2$

respectively. By abusing notation a little bit we write :

$$\check{l}_i \sim \mathcal{N}\Big([\boldsymbol{G\mu}]_i, \frac{\check{y}_i^{obs}}{(\check{y}_i^{obs} - r_i)^2}\Big) \tag{2.3}$$

One could also use elementary probability theorems and simplifying assumptions to arrive at the above approximate distribution. Most of the algorithms in this thesis are based on either (2.2) or (2.3).

2.4 Choosing a system model

$\boldsymbol{G}$ models the integral operation over the part of the image covered by a ray that is passing from the source to the detector. It is one of the key components of a statistical method – the choice of the system model affects both image quality and reconstruction time. The operator G is also called the forward-projection operation and is computationally intensive. The various approaches to implementing a system-model can be divided into three categories : direct [13], Fourier [70] and hierarchical [6, 7]. Direct approaches (distance-driven forward and backprojectors [13]) have been used in the current work due to the availability of a high speed numerical implementation from our collaborators at General Electric Healthcare Inc.. Blob-based system models for SPECT have been investigated in [68]. Finite-sized sources can be modeled by modifying $\boldsymbol{G}$ by post-multiplying it by a sparse square matrix of size $n_d \times n_d$; $\boldsymbol{G}$ is said to be in a factored form when this post-multiplication is performed. Having $\boldsymbol{G}$ in a factored form causes the OS method to become sub-optimal [2]. This necessitates the use of PCG-based algorithms. Other physical processes that could possibly be accounted for using the system model are afterglow [32] and scatter [69]. A preliminary analysis points to a factored system matrix in these cases also.

[2]Whether OS methods will still work in presence of factored system matrices needs to be checked using experiments.

Storage of the entire matrix G is not possible, even in a sparse format. The domain and range sizes of G for a medium-sized helical cone beam reconstruction problem are both in the range of 200 million. After sparsity considerations, the number of non-zero entries in G would be of the order of 100 billion and would occupy atleast 1 terabyte of memory. Computation and storage of G is time consuming and requires excessive computer resources. Thus, the entries of G are computed when the need to access them arises during computation.

2.5 Including physical effects like beam hardening

Beam hardening is the phenomenon in which the mean energy of the photons of an X-ray beam increases as the beam progresses through the body. This happens because the materials that make up the human body attenuate lower energy photons more than higher energy photons. This causes the Beer's Law of (2.1) to no longer hold, and has to be replaced as shown below. Neglecting beam hardening in images in two types of artifacts: cupping and streaks. Some form of beam hardening correction is necessary if an accurate measurement of the X-ray attenuation coefficient is desired. A general introduction to the interaction of X-ray photons with matter and beam hardening can be found in [46].

In current clinical CT practice, most physical effects are corrected for using sinogram pre-correction techniques, which are usually deterministic methods. The JS method is the preferred method for beam hardening sinogram pre-correction. However, with reducing dose, inclusion of the physical effects into the model and the cost function becomes necessary. In the rest of this section, the distribution of the observation that includes the beam hardening effect is derived.

The Poisson model is modified to include the beam-hardening effect as follows [20] :

$$y_i \sim \mathsf{Poisson}\left\{ \int I_i(\mathcal{E}) e^{-\int_{L_i} \mu(\boldsymbol{r},\mathcal{E}) d\boldsymbol{r}} d\mathcal{E} + r_i \right\},$$

where, $I_i(\mathcal{E})$ is the energy spectrum of the incident X-rays, $\mu(\boldsymbol{r}, \mathcal{E})$ is the X-ray attenuation coefficient for energy $\mathcal{E}$ at location $\boldsymbol{r}$ in the object. The attenuation coefficient is the product of mass attenuation coefficient $m(\mathcal{E})$ and the material density $\rho(\boldsymbol{r})$ [46] :

$$\mu(\boldsymbol{r}, \mathcal{E}) = m(\mathcal{E})\rho(\boldsymbol{r}).$$

Note that m and ρ depend only on $\mathcal{E}$ and $\boldsymbol{r}$ respectively.

The tissues within the body can be divided into two types, soft-tissue and bone, on the basis of variation of $m(\mathcal{E})$ with respect to $\mathcal{E}$. The mass-attenuation coefficient of the various types soft-tissues like muscle, fat, breast *etc.*, are similar to water. Therefore, the terms water and soft-tissue are used interchangeably. This simplistic classification into two classes removes the most apparent beam hardening artifacts, but it does not yield accurate CT numbers, especially in single-energy abdominal scans [54]. Dual-energy CT has been shown to quantify fat content more accurately [54, 67], but has not yet replaced single-energy CT, which continues to be the standard. The field of X-ray CT technology is moving towards the implementation of dual-energy CT by developing new detectors, X-ray tubes, and scanning methods.

The X-ray attenuation at a point can be written as the sum of attenuation due to the two substances as follows:

$$\mu(\boldsymbol{r}, \mathcal{E}) = \mu_S(\boldsymbol{r}, \mathcal{E}) + \mu_B(\boldsymbol{r}, \mathcal{E}) = m_S(\mathcal{E}) \cdot f_S(\boldsymbol{r})\rho(\boldsymbol{r}) + m_B(\mathcal{E}) \cdot f_B(\boldsymbol{r})\rho(\boldsymbol{r})$$

$\rho(\boldsymbol{r})$ is the total material density at location $\boldsymbol{r}$. $f_S(\boldsymbol{r})$ is the relative amount of soft tissue present at point $\boldsymbol{r}$ and its value varies from 0 to 1. Values of $f_S(\boldsymbol{r})$ and $f_B(\boldsymbol{r})$ can be found by image segmentation [20] or can be fixed as a function of density $\rho(\boldsymbol{r})$ [21]. The latter approach is possible because the densities of soft-tissues like brain, muscle and lung are close to water, *i.e.*, 1.0 g/cc and the density of bone tissue is close to 1.9 g/cc. The images currently being produced are at such a coarse scale (about a millimeter) and the structure of bone tissue is such that the number of pixels involving purely the bone tissue is small. As a consequence, many pixels are partly bone and partly soft-tissue and image quality is influenced by how accurately the fractions are obtained. Thus, the distribution of the observations that incorporate the beam-hardening effect can be written as :

$$\begin{aligned} y_i &\sim \mathsf{Poisson}\left\{b_i e^{-f_i(T_{S,i},T_{B,i})} + r_i\right\}, \\ f_i(T_{S,i},T_{B,i}) &\stackrel{\triangle}{=} -\ln \int \frac{I_i(\mathcal{E})}{b_i} e^{(-m_S(\mathcal{E})T_{S,i} - m_B(\mathcal{E})T_{B,i})} d\mathcal{E}, \\ b_i &\stackrel{\triangle}{=} \int I_i(\mathcal{E}) d\mathcal{E}, \\ T_{S,i} &\stackrel{\triangle}{=} \int_{L_i} f_S(\boldsymbol{r})\rho(\boldsymbol{r}) d\boldsymbol{r} \approx [\boldsymbol{G} I_S \boldsymbol{\rho}]_i, \\ T_{B,i} &\stackrel{\triangle}{=} \int_{L_i} f_B(\boldsymbol{r})\rho(\boldsymbol{r}) d\boldsymbol{r} \approx [\boldsymbol{G} I_B \boldsymbol{\rho}]_i, \end{aligned} \tag{2.4}$$

$\boldsymbol{\rho}$ is the vector of density values of the image and I_S and I_B are diagonal matrices containing the pixelized values of f_S and f_B. [3] We use this model in Chapter 4.

2.6 Formulating a cost function

The cost function consists of the negative log-likelihood function, which is computed from the statistical models obtained above. A penalty function or a Bayesian prior on the image is added to the likelihood depending on additional requirements. Additional requirements include statistical priors on the image, edge preservation, increased noise reduction, non-negative image pixels, space-invariant image resolution and space-invariant image noise. The penalty function also assists in faster convergence. It has been observed that

[3]The unit of ρ is g/cc. The value is easily converted to HU by multiplying it by 1000.

the condition number of the problem is reduced by regularization, thus speeding convergence. The negative log-likelihood for the Poisson and WLS cases with beam-hardening pre-corrected is :

$$-L(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{G\mu}]_i), \tag{2.5}$$

$$h_i(t_i) = \begin{cases} \bar{y}_i - \check{y}_i \ln(\bar{y}_i), & \bar{y}_i = b_i e^{-t_i} + r_i, \quad \text{Poisson}, \\ \frac{1}{2}\check{y}_i(t_i - \check{l}_i)^2, & \qquad\qquad\qquad\quad \text{WLS}. \end{cases}$$

When beam-hardening is taken into account in the Poisson model, the negative log-likelihood is :

$$-L(\boldsymbol{\rho}) = \sum_{i=1}^{n_d} h_i(\bar{y}_i), \tag{2.6}$$

$$h_i(\bar{y}_i) = \bar{y}_i - y_i \ln(\bar{y}_i),$$

$$\bar{y}_i = b_i \exp(-f_i([\boldsymbol{G}I_S\boldsymbol{\rho}]_i, [\boldsymbol{G}I_B\boldsymbol{\rho}]_i)) + r_i,$$

where, f_i was defined in (2.4).

2.6.1 Regularization

A penalty function or regularization function is added to the negative log-likelihood and is usually of the form :

$$R(\boldsymbol{\mu}) = \sum_{k=1}^{n_K} \psi_k([\boldsymbol{C\mu}]_k). \tag{2.7}$$

The main goals of the penalty function are to reduce noise and preserve edges. This is done by discouraging the differences between neighboring image pixels from becoming too large. The difference between the pixels of the kth pixel pair are denoted by $[\boldsymbol{C\mu}]_k$. ψ_k is called the potential function and its form denotes the amount of regularization for a particular value of the pixel difference. Choice of ψ_k influences the reconstructed image.

	$\psi(t)$	$\omega_\psi(t)$
Quadratic	$t^2/2$	1
Huber	$\begin{cases} t^2/2 & \lvert t\rvert \le \delta \\ \delta\lvert t\rvert - \delta^2/2 & \lvert t\rvert > \delta \end{cases}$	$\begin{cases} 1 & \lvert t\rvert \le \delta \\ \delta/\lvert t\rvert & \lvert t\rvert > \delta \end{cases}$
Hyperbola	$\delta^2[\sqrt{1+(t/\delta)^2}-1]$	$\frac{1}{\sqrt{1+(t/\delta)^2}}$
q-GGMRF [61]	$\frac{\lvert t\rvert^p}{1+\lvert t/\delta\rvert^{p-q}}$	$\frac{\lvert t\rvert^{p-2}}{1+\lvert t/\delta\rvert^{p-q}}(p-(p-q)\frac{\lvert t/\delta\rvert^{p-q}}{1+\lvert t/\delta\rvert^{p-q}})$
Generalized Fair [24]	$\frac{\delta^2}{2b^3}(2b(b-a)\lvert t/\delta\rvert$ $+ab^2\lvert t/\delta\rvert^2$ $+2(a-b)\log(1+b\lvert t/\delta\rvert))$	$\frac{1+a\lvert t/\delta\rvert}{1+b\lvert t/\delta\rvert}$

Table 2.1: Expressions for potential functions, $\psi(t)$, and corresponding weighting functions, $\omega_\psi(t)$.

For example, when ψ_k is a quadratic function the edges in the reconstructed image are smooth. In order to prevent this, a non-quadratic function that is not too large for large values of the pixel differences, like the Huber function, is used [34]. The potential functions used and investigated in this thesis are listed in Table 2.1. The field of regularization design involves choosing a penalty function to meet a specific aim. It is inspired by a large number of fields like robust statistics, total variation methods, and diffusion equations.

The parameters of the penalty function (embedded inside ψ_k) could be space-variant or space-invariant. Space-invariant parameters mean that β and other parameters of non-quadratic regularization, *e.g.*, δ in Huber potential function, do not depend on the location of the pixels. It has been shown in [26] that space-invariant penalties can lead to a space-variant local impulse response; some areas of the reconstructed image are more blurred than the others. The penalty function suggested in [26] adjusts the penalty function parameters so that a near uniform impulse response can be achieved. This penalty function is referred to as the space-variant penalty. The preconditioners of PCG algorithms are different when different penalties are used (see Section 2.7.3).

Thus, the overall cost function is :

$$\Phi(\boldsymbol{\mu}) = -L(\boldsymbol{\mu}) + \beta R(\boldsymbol{\mu}).$$

β is a positive parameter that decides how “strong” the regularization is. If the observed data is relatively noise-free then the preferred value of β would be small and vice versa. The method of choosing β is to first create reconstructions at different β values, and then choose the β according to some criterion. Various criteria like L-curves, cross-validation, local-resolution and bias-variance trade-off have been suggested in literature. The β so chosen would not only depend on the noise-level of the observations (*i.e.*, initial source intensity) but also on the rough size of the patient. So, a table of β values will have to be created during the calibration of the scanner and its reconstruction software.

2.6.2 Cost function properties

The cost function constructed above has to be minimized in order to compute the reconstructed image. A few questions have to be addressed at this point :

- Does a minimum exist?
- Do images close to the minimum or images with sufficiently low cost function value possess satisfactory image properties?
- If a minimum exists, is it unique?
- Are minima of the cost function global or local?
- Do local minima possess satisfactory image properties?

Such questions can be answered by mathematical arguments by using theorems from optimization theory and checked, at least partially, by experiments. The above questions are a bit more difficult to answer in the constrained case than in the unconstrained case. When the reconstructed image is constrained to lie within a target set, *e.g.* when the pixels of the image are required to be positive, the minimization (or optimization) is said to be constrained. Such questions can be posed with respect to the continuous[4] image $\mu(\boldsymbol{r})$ instead

[4]Continuous in the sense of the domain being continuous as opposed to discrete or pixelized.

of its approximation $\boldsymbol{\mu}$. Some analysis tools consider $\mu(\boldsymbol{r})$ as a distribution rather than a continuous image. This view aids in a easier handling of edges which are then referred to as singularities [53]. In this case, analysis requires the slightly more involved concepts from functional analysis. Analysis using $\boldsymbol{\mu}$ is easier due to the use of concepts from linear algebra *e.g.* null spaces, matrix rank *etc.*. The cost function may not be differentiable due to certain potential functions in the regularization not being differentiable (*e.g.* broken parabola) and analysis for non-differentiable functions has to be used. Theoretical answers to such questions allow us to predict the properties of the reconstructed image and the statistical reconstruction method. Sometimes such theoretical questions are of lesser importance than the more practical goal of producing an image of acceptable visual quality in an acceptable amount of time. The above acceptability is with respect to the end-users of the applications *e.g.* the radiologists.

In summary, the reconstructed image, $\hat{\boldsymbol{\mu}}$, is related to the cost function, $\Phi(\boldsymbol{\mu})$ as follows :

$$\hat{\boldsymbol{\mu}} = \begin{cases} \text{arg min}_{\boldsymbol{\mu}} \Phi(\boldsymbol{\mu}), & \text{(Unconstrained optimization)}, \\ \text{arg min}_{\boldsymbol{\mu} \in S} \Phi(\boldsymbol{\mu}),\ S \triangleq \{\boldsymbol{\mu} \in \mathcal{R}^{n_p} : & \forall j = 1, \ldots, n_p, \mu_j \geq 0\} \\ & \text{(Non-negatively constrained optimization)}. \end{cases}$$

Depending on factors like the trade-off between accuracy and speed, an unconstrained or a constrained optimization algorithm is used.

2.7 Choosing a minimization algorithm

The cost function has to be minimized in order to produce a reconstruction. One step solutions to minimization of cost functions are rare. Instead, one starts with a guess, $\boldsymbol{\mu}^{(0)}$, and proceeds to refine it over and over, attempting to reduce the value of the cost function at every step. The numerical methods used to minimize the cost functions, known as algorithms, are usually based on gradients. Sometimes non-differentiable cost functions can be approximated by differentiable functions and a reconstruction computed. Two favorable properties that an algorithm could possess are monotonicity and convergence. An algorithm is said to be convergent if there exists an image to which the iterates get arbitrarily close to. An algorithm is said to be monotonic if each newer iterate achieves a cost function value that is lesser than or equal to the cost function value of the current iterate. Most algorithms have a stopping or convergence criteria. For example, if the change in image pixels or the change in cost function values become too small then the algorithm is stopped. Sometimes visual quality experienced by an end-user is used as a stopping criteria. Due to the finite word-length and memory of the computers used, algorithms can not proceed indefinitely to the theoretical minimum of the cost function and have to be stopped.

The initial guess is usually an image from a traditional method like FBP. If the cost function properties do not guarantee convergence to a single global minimum, *e.g.*, if it has local minima or is non-convex, then a good initial guess is necessary. For convex cost functions with a unique minimum, a uniform image of zeros suffices as a initializer. But, the minimization usually takes a larger number of iterations to converge/stop when starting from a uniform zero image rather than the FBP image, because the former is usually farther from the minum/minima of the cost function.

The rest of the subsections in this section are organized as follows. The method of optimization transfer in deriving algorithms is described in Section 2.7.1. Various techniques are suitable to accelerate algorithm convergence for cost functions seen in X-ray CT:

1. Ordered subsets (OS) (Section 2.7.2),
2. Preconditioned conjugate gradient (PCG) (Section 2.7.3), and
3. Incremental Coordinate Descent (ICD) (Section 2.7.4).

Non-negatively constrained algorithms are described in Section 2.7.5. Finally, the properties of various algorithms are summarized in Section 2.7.6.

2.7.1 Iterative minimization using optimization transfer

A surrogate of the cost function is a function whose minimization leads to the minimization or reduction of the value of the cost function. Surrogate based optimization algorithms are also referred to as optimization transfer methods or MM algorithms [36]. The surrogate offers certain advantages like making the function being minimized quadratic and guaranteeing monotonicity. Surrogates used to be computed by statistical arguments and resulted in algorithms called EM algorithms. EM algorithms have been included in a larger set of surrogates that include quadratic functions. The quadratic surrogates afford faster convergence than EM algorithms and allow the possibility of using unconstrained and constrained algorithms already invented in the field of quadratic optimization.

Let our initial guess be $\boldsymbol{\mu}^{(0)}$, and the iterates computed by an algorithm be $\boldsymbol{\mu}^{(n)}, n = 0, 1, 2, \ldots$. The surrogate function, $\phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)})$, of the cost function, $\Phi(\boldsymbol{\mu})$, is a function that is computed at the nth iterate. The surrogate function has the property that reducing its value, reduces the value of the cost function, *i.e.*, $\phi(\boldsymbol{\mu}^{(n+1)}; \boldsymbol{\mu}^{(n)}) \leq \phi(\boldsymbol{\mu}^{(n)}; \boldsymbol{\mu}^{(n)}) \Rightarrow \Phi(\boldsymbol{\mu}^{(n+1)}) \leq \Phi(\boldsymbol{\mu}^{(n)})$. Thus, reducing the value of the surrogate function guarantees a monotonic reduction in the value of the cost function. The following conditions are sufficient for a function $\phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)})$ to be a surrogate of a cost function $\Phi(\boldsymbol{\mu})$ at the nth iterate [44, Eq. 2] :

$$\phi(\boldsymbol{\mu}^{(n)}; \boldsymbol{\mu}^{(n)}) = \Phi(\boldsymbol{\mu}^{(n)}), \tag{2.8}$$

$$\phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) \geq \Phi(\boldsymbol{\mu}), \ \forall \boldsymbol{\mu} \in \mathbb{R}_p^n. \tag{2.9}$$

When $\phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)})$ and $\Phi(\boldsymbol{\mu})$ are differentiable, the gradients of the surrogate function and the original cost function at the current iterate $\boldsymbol{\mu}^{(n)}$ shall be equal :

$$\frac{\partial \phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)})}{\partial \mu_j}\Big|_{\boldsymbol{\mu}=\boldsymbol{\mu}^{(n)}} = \frac{\partial \Phi(\boldsymbol{\mu})}{\partial \mu_j}\Big|_{\boldsymbol{\mu}=\boldsymbol{\mu}^{(n)}}, j = 1, \ldots, n_p. \tag{2.10}$$

When a function of the image is of the form $\Phi(\boldsymbol{\mu}) = \sum_{i=1}^{n'_d} h'_i([\boldsymbol{A\mu}]_i)$ then it is said to be additively-separable. A quadratic surrogate for the above function can be determined if we majorize each of the terms using :

$$h'_i(t) \leq q'_i(t; s) \triangleq h'_i(s) + \dot{h}'_i(s)(t - s) + \tfrac{1}{2}\breve{c}'_i(s)(t - s)^2 \text{ and,} \tag{2.11}$$

$$\phi(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) \triangleq \sum_{i=1}^{n'_d} q'_i([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$$

Determining the curvatures $\breve{c}'_i(s)$ is the crucial computation in the above equation because it influences the convergence rate to a large extent. In general, the smaller the value of $\breve{c}'_i(s)$, the larger will be the step taken by an algorithm, and faster will be the convergence. One of the methods to compute curvatures for the negative log-likelihood part is shown in [1], and for the regularization part in [34].

2.7.2 Ordered subsets (OS)

Ordered subsets or incremental gradient is a method to approximately compute the gradient using only a subset of the observed data. A set of subsets of the observed data is fixed and the gradient is obtained in a cyclic order from each of them. If each subset provides nearly the same gradient then the condition is called subset balance. This condition is vital for the OS approach to work. Using a limited amount of data results in significant computational savings. Recall from Section 2.6 that the cost function is usually of the form :

$$\Phi(\boldsymbol{\mu}) = -L(\boldsymbol{\mu}) + \beta R(\boldsymbol{\mu}), \qquad -L(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{G\mu}]_i), \qquad R(\boldsymbol{\mu}) = \sum_{k=1}^{n_K} \psi_k([\boldsymbol{C\mu}]_k).$$

The OS approximation is used usually for $-L(\boldsymbol{\mu})$. The observed data set $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_d}\}$ is divided into M subsets and $-L(\boldsymbol{\mu})$ can be expressed as : $-L(\boldsymbol{\mu}) = \sum_{m=1}^{M} \Phi_m(\boldsymbol{\mu})$. Thus, $\nabla(-L(\boldsymbol{\mu})) = \sum_{m=1}^{M} \nabla\Phi_m(\boldsymbol{\mu})$. By the subset-balance condition, $\nabla\Phi_1(\boldsymbol{\mu}) \approx \nabla\Phi_2(\boldsymbol{\mu}) \approx \cdots \approx \nabla\Phi_M(\boldsymbol{\mu})$. Thus, $\nabla(-L(\boldsymbol{\mu})) \approx M\nabla\Phi_m(\boldsymbol{\mu})$. The gradients computed using OS are observed to be good approximations only when the iterates are far away from the minimum. Thus, OS is not suited for stringent convergence criteria. A convergent version of OS called incremental optimization transfer (IOT) was recently proposed [3] but it suffers from large memory requirements. The memory requirements of IOT may be too large for the 3-D helical cone-beam CT application considered in Chapter 7 .

A recent and related OS algorithm is the incremental aggregated gradient (IAG) [9]. A distributed implementation of IAG is also described in the above paper. When IAG is adapted to the context of X-ray CT, it turns out that the memory requirement of IAG are atleast one image volume per subset. This memory requirement is so large that IAG becomes impractical for X-ray CT.

2.7.3 Preconditioned conjugate gradient (PCG)

When minimizing a differentiable cost function or its quadratic surrogate, it is necessary to compute the gradient. However, the gradient direction is not the best search direction. For example, a 2-argument unimodal function with highly elliptical contours has a gradient at most points in the domain that points in a direction away from the minimum. In the conjugate gradient method, a better descent direction is produced by combining the gradient at the current iterate with the descent direction at the previous iterate. The basis for such a combination is in the theory of Krylov space methods. The Krylov space methods are applicable here because minimizing a quadratic function is equivalent to solving a system of equations. This is another motivation for using quadratic surrogates. Numerous methods exist even in the conjugate gradient category and one such is outlined below.

Preconditioning is incorporated into the conjugate-gradient method to produce the PCG method. A preconditioner is an approximation to inverse of the Hessian or an easily invertible matrix that is an approximation to the Hessian. Preconditioning transforms the variables so as to cause the iterates to converge faster by improving the condition number of the problem. One notes here that if the preconditioner P is the exact inverse of the Hessian of a quadratic surrogate then the iteration would converge in one step by using the Newton's method. One major advantage of PCG over OS is that any approximation to the inverse of the Hessian can act as a preconditioner as long as it is positive definite. A monotonic algorithm will be obtained as long as we have a valid preconditioner. This is in contrast to the OS based methods where approximate gradient computation results in a non-monotonic algorithm.

The PCG method computes the descent direction by first computing the gradient. The gradient is multiplied by a preconditioner matrix to obtain an intermediate vector, $\boldsymbol{p}^{(n)}$. The previous descent direction is weighted by a factor $\gamma^{(n)}$ and added to the negative of the intermediate vector to obtain the final descent direction. The whole procedure is summarized as follows :

$$\begin{aligned}
&\boldsymbol{g}^{(n)} = \nabla\Phi(\boldsymbol{\mu})\big|_{\boldsymbol{\mu}=\boldsymbol{\mu}^{(n)}}, && \text{Gradient,}\\
&\boldsymbol{p}^{(n)} = \boldsymbol{P}\boldsymbol{g}^{(n)}, && \text{Pre-conditioned gradient,}\\
&\gamma^{(n)} = \begin{cases} 0 & n = 0, \\ \frac{\boldsymbol{p}^{(n)T}(\boldsymbol{g}^{(n)}-\boldsymbol{g}^{(n-1)})}{\boldsymbol{p}^{(n-1)T}\boldsymbol{g}^{(n-1)}} & n > 0, \end{cases} && \text{Polak-Ribiere formula,}
\end{aligned}$$

(2.12)

$$\boldsymbol{d}^{(n)} = -\boldsymbol{p}^{(n)} + \gamma^{(n)}\boldsymbol{d}^{(n-1)}, \qquad \text{Pre-conditioned conjugate gradient direction.}$$

Once the descent direction for either the cost function or its surrogate is determined a step size is chosen to compute the new iterate as follows :

$$\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \alpha^{(n)}\boldsymbol{d}^{(n)} \tag{2.13}$$

$$\text{where, } \alpha^{(n)} = \text{arg min}_{\{\alpha\in\mathcal{R}:\alpha\geq 0\}}\Phi^{(n)}(\boldsymbol{\mu}^{(n)} + \alpha\boldsymbol{d}^{(n)})$$

This is called a line search. For a quadratic surrogate, this step can be done analytically. For a non-quadratic function a surrogate over α is computed [25]. Sometimes, the De-Pierro's trick [15] is employed to create a surrogate in which the individual pixels are separated from each other. This permits us to create a pixel update and such a surrogate is referred to as a separable surrogate.

2.7.4 Iterative Coordinate Descent (ICD)

The iterative coordinate descent algorithm reduces the value of the cost function one voxel at a time [56, 61]. At a given iterate, all voxels except one are held constant. A new value of the variable voxel that reduces the cost function value is computed. The next voxel shall be updated using the newly computed value of the current voxel. This process is repeated with all voxels, over and over, till convergence criteria are satisfied.

ICD is a convergent algorithm with a high convergence rate, which makes it a near-ideal algorithm. However, it has certain drawbacks, one of which we try to overcome using the hybrid OS-ICD algorithm in Section 7.1.2. ICD has the property that the high frequency components of the image converge at a high convergence rate, whereas the low frequency components have a low convergence rate. In some cases, due to the greedy nature of the algorithm and its high-convergence rate, the voxel values overshoot the minimum/minima, taking a large number of iterations to converge to the solution. This happens when the difference between the current iterate and the solution is large. Another drawback is that its implementations on general-purpose microprocessors have memory bottlenecks. Due to voxel based nature of the algorithm, the forward and back projector implementations have to be voxel based. Voxel-based projector implementations have a highly non-sequential memory access, causing the microprocessor cache to become useless. Caches are vital in modern general-purpose microprocessors because the microprocessors are faster than the random access memory. Without cache utilization code execution slows down. OS and PCG based methods do not suffer from this drawback because their implementations of forward and back projectors have more sequential memory access patterns.

2.7.5 Non-negatively constrained algorithms

The algorithms used for constrained (more specifically, non-negatively constrained) and unconstrained minimization differ a great deal in PCG based algorithms. This is because the non-negativity constraint makes the line search (2.13) complicated. OS methods usually require the creation of a separable quadratic surrogate, which makes the application of non-negativity constraint trivial. Thus, OS methods are better suited to the application of the non-negativity constraint than the PCG based methods. The non-negativity constraint can be applied easily to ICD based methods also because of their voxel-based updates.

Numerous methods for applying the non-negativity constraint to gradient based methods have been described in literature. For example, LBFGS-B algorithm [10] and a method for handling non-negative components in conjugate direction methods in [28]. Numerous applications of LBFGS-B algorithm have been investigated, but it has not yet been proven to be useful for 3-D reconstructions of MSCT data. Other examples are, an interior-point method for PET [37], and a non-negatively constrained CG method for astronomical imaging [5]. All of the above methods are generic in nature, as they use the quasi-Newton preconditioner.

	OS	PCG	ICD
Monotonicity	No	Yes	Yes
Convergence rate (low frequencies)	High	Medium	Low
Convergence rate (high frequencies)	High	Medium	High
Memory access	Sequential	Sequential	Non-sequential
Application of non-negativity constraint	Easy	Hard	Easy
Factored system matrix	No	Yes	Yes

Table 2.2: Properties of candidate algorithms for X-ray CT.

2.7.6 Summary of algorithm properties

OS, PCG, and ICD are candidate algorithms for use in statistical image reconstruction methods in X-ray CT. Their properties are summarized in Table 2.2. From the table, it is evident that one single algorithm does not contain all the desirable features. Therefore, further development of new algorithms is a necessity.

2.8 Cone-beam geometry

A 2-D fan-beam X-ray scanner has a single source and a single row of detectors (Fig. 2.1) and it collects sinograms for only one slice, in one rotation of the gantry. Multiple rotations of the gantry and bed positions have to be used to scan multiple slices of the patient anatomy. The cone-beam geometry is created from the fan-beam geometry by adding rows of detectors (in the axial-direction) on either side of the central row of detectors (see Fig. 2.2). This allows multiple slices to be scanned in one rotation, hence the name, Multislice CT (MSCT) scanner. If there is no bed-motion then the source traces a circular orbit around the patient. Such a data-collection geometry is called the axial cone-beam geometry. If there is bed motion at a constant speed along with the gantry rotation, then the source moves in a helical path around the patient and the data-collection geometry is called the helical cone-beam geometry. Specialized applications of MSCT

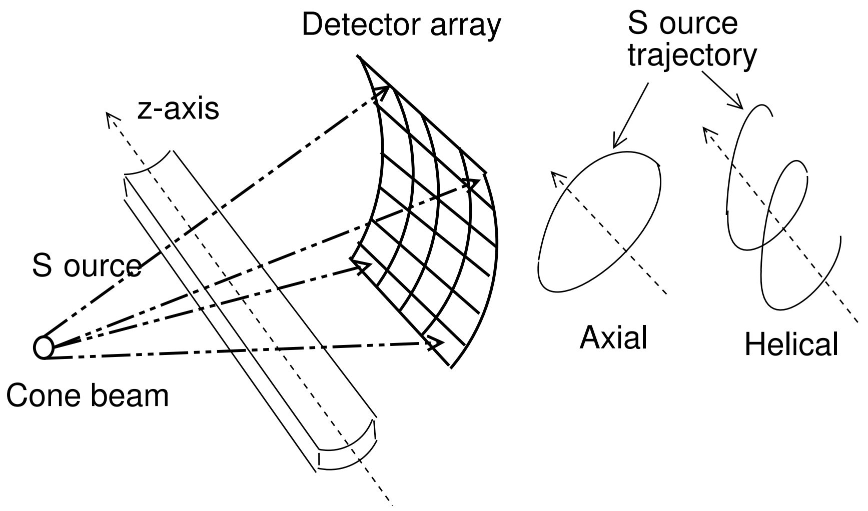


Figure 2.2: Axial and helical cone beam X-ray CT.

scanners *e.g.*, cardiac, create a more complex data-collection geometry by choosing data from axial or helical scans using an external signal, *e.g.*, electro-cardiogram (ECG). The reconstruction methods should be flexible enough to reconstruct images from all the above data-collection geometries.

Not all geometries can be used to reconstruct the region-of-interest correctly; a necessary and sufficient condition for a geometry to be useful was given in [58]. The above condition (as quoted in [47]) is

If on every plane that intersects the object, there exists at least one cone beam source point, then one can reconstruct the object.

Another similar condition is the PI-sufficiency condition (as quoted in [52]) is

The so-called PI-sufficiency condition requires that each point must be illuminated by the source over an angular span of π, as seen from the point.

According to both the above conditions, it is apparent that off-center slices in axial cone-beam geometry cannot be reconstructed correctly. Helical geometry satisfies the above conditions if the object lies within the helix. But, if the object extends beyond the size of the helix then the above conditions are not satisfied leading to the long object problem. Various solutions to the long object problem have been suggested in [16,40,60].

Various analytical methods have been proposed to reconstruct the object from its 3D projections for both axial and helical cone-beam geometries. Some methods are theoretically exact (based on sound mathematics) while others are approximate (use intuition or a simplifying observation). A few of the exact methods are due to Grangeat [27], Tuy [64], Tam [60] and Katsevich [40]. Most exact methods are computationally intensive except [40]. Approximate methods typically require much less computation and produce satisfactory images along with a small amount of artifact. Computational savings in approximate methods are typically produced by deriving FBP-like methods *i.e.* methods involving one or more of weighting, rebinning, shift-invariant filtering and 3D or 2D back-projection. A well-known approximate algorithm for axial cone-beam geometry is the FDK algorithm [22]. The algorithm computes the increment to a voxel due to the source being located at a particular angle by assuming an equivalent infinitesimal circular rotation in the tilted plane (the plane of the source and a row of off-center detectors). Among the plethora of approximate methods the notable ones are the PI-method [63] and the wedge-beam method of [65]. The PI-method has been expanded in [52] to include the case when the pitch of the helix is small when compared to the detector-size.

To compute an initializer for iterative reconstruction, an approximate method such as FBP is sufficient. This is because the iterative reconstruction algorithms shall correct the artifacts produced due to the inexact analytical reconstruction methods also, apart from their regular functions like noise reduction.

CHAPTER 3

Cost function for the non-negativity constraint

A basic property of the X-ray linear attenuation coefficient is that it takes only non-negative values. FBP and other conventional analytical methods do not take this fact into account during the inversion process. These methods impose the non-negativity constraint as an after thought leading to a sub-optimal reconstruction. In contrast to FBP, statistical methods that use constrained optimization algorithms allow the non-negativity constraint to be imposed during the inversion process.

As described in Section 2.7, the requirement of low computation time in statistical methods leads us to use OS and PCG based algorithms. The advantage of OS over PCG based algorithms is that the imposition of the non-negativity constraint can be done without any compute time overhead. Whereas, in the latter the overhead is 50% (see Section 3.1). However, incorporation of various physical effects cause the OS methods to become sub-optimal (see Section 2.4). Moreover, OS has to be replaced by PCG when monotonicity is desired. This is because monotonicity can be achieved much more easily in PCG when compared to OS. Therefore, reducing the compute time overhead of PCG based algorithms is essential if non-negative reconstructions are desired in a reasonable amount of time. In our solution, we abandon the explicit imposition of the non-negativity constraint and let regularization control the negative pixels. In order to control the negative pixels further, we modify the negative log-likelihood (in a manner that bears some likeness to penalty based constrained optimization [45]). The resulting algorithm is unconstrained, monotonic (in the modified cost function) and controls negative pixels. The method developed in this chapter is for the mono-energetic Poisson case but it can be easily extended to the poly-energetic case.

A general non-negatively constrained image estimate was defined Section 2.7.5, and is repeated here for convenience :

$$\hat{\boldsymbol{\mu}} = \text{arg min}_{\boldsymbol{\mu} \in S} \Phi(\boldsymbol{\mu}),$$
$$\Phi(\boldsymbol{\mu}) = -L(\boldsymbol{\mu}) + \beta R(\boldsymbol{\mu}), \quad -L(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{G}\boldsymbol{\mu}]_i), \quad R(\boldsymbol{\mu}) = \sum_{k=1}^{n_K} \psi_k([\boldsymbol{C}\boldsymbol{\mu}]_k),$$

where, $S \triangleq \{\boldsymbol{\mu} \in \mathcal{R}^{n_p} : \forall j = 1, \ldots, n_p, \mu_j \geq 0\}$, is the non-negative orthant. For the mono-energetic Poisson case we have,

$$h_i(t) = b_i e^{-t} + r_i - \check{y}_i \ln(b_i e^{-t} + r_i). \tag{3.1}$$

After the cost function modification (described below), the image estimate would be obtained through unconstrained optimization as

$$\tilde{\boldsymbol{\mu}} = \text{arg min}_{\boldsymbol{\mu}} \tilde{\Phi}(\boldsymbol{\mu}),$$
$$\tilde{\Phi}(\boldsymbol{\mu}) = -\tilde{L}(\boldsymbol{\mu}) + \beta R(\boldsymbol{\mu}), \quad -\tilde{L}(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} \tilde{h}_i([\boldsymbol{G}\boldsymbol{\mu}]_i), \quad R(\boldsymbol{\mu}) = \sum_{k=1}^{n_K} \psi_k([\boldsymbol{C}\boldsymbol{\mu}]_k).$$

Ideally, one would like to achieve $\tilde{\boldsymbol{\mu}} \approx \hat{\boldsymbol{\mu}}$.

3.1 Increase of compute time in PCG algorithms due to the non-negativity constraint

We consider the cases of monotonic and non-monotonic PCG algorithms separately.

A monotonic PCG algorithm must use quadratic surrogates based on optimal curvatures [1, PSOC algorithm] in order to monotonically minimize the cost function. But, the PSOC

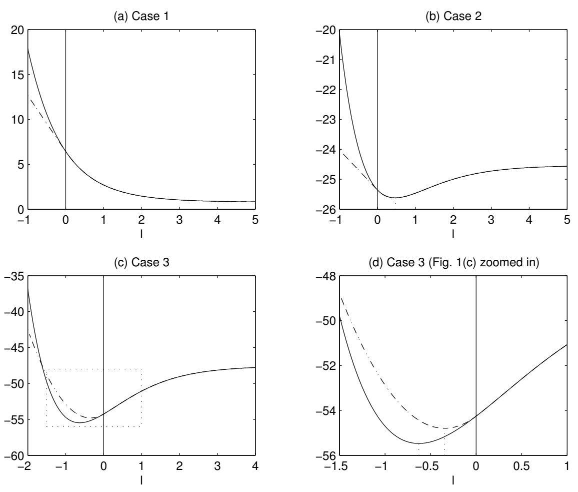


Figure 3.1: Plots of $h_i(t)$ (solid line) and $\tilde{h}_i(t)$ (dashed line) for the three categories. $r_i = 10$, $b_i = 8$, and $y_i = 4, 15$ and 25 in (a), (b) and (c) above respectively. Note that $\tilde{h}_i = h_i$ for $t \geq 0$ and differs only for non-physical values of the attenuation coefficients.

algorithm requires the non-negativity constraint to be enforced on the image at every step in order to ensure monotonicity. Due to the requirement of the imposition of the non-negativity constraint in PSOC the line search of (2.13) has to be changed to :

$$\boldsymbol{\mu}^{(n+1)} = [\boldsymbol{\mu}^{(n)} + \alpha^{(n)} \boldsymbol{d}^{(n)}]_+. \tag{3.2}$$

The use of line search of (2.13) allows one less forward projection operation per iteration in the implementation. Thus, using line search of (3.2) raises the computational load by about 50%.

Non-monotonic PCG algorithms use quadratic surrogates based on the pre-computed curvatures in order to increase the convergence rate. These algorithms have to use the line search of (3.2) instead of (2.13) too in order to impose non-negativity. Thus, there is an overhead of 50% in this case too.

3.2 Cost function modification

In this section the intuition behind the cost function modification is discussed. This is followed by the definition of the modified likelihood. Next, the quadratic surrogate for the modified likelihood is presented. The proof of majorization of the quadratic surrogate over the modified cost function can be found in Section A.2.2.

The intuition behind the modification of the likelihood is as follows. The negative log likelihood is a sum of functions, h_i, that depend on the values of b_i, r_i and the measurements $\check{y}_i$. The arguments of these functions are $[\boldsymbol{G}\boldsymbol{\mu}]_i$. If $[\boldsymbol{G}\boldsymbol{\mu}]_i$ is negative then at least one of the μ_j's is negative. This is because the elements of the matrix $\boldsymbol{G}$ are non-negative. To state more concisely,

$$[\boldsymbol{G}\boldsymbol{\mu}]_i < 0 \quad \Rightarrow \quad \exists\, j \text{ such that } \mu_j < 0. \tag{3.3}$$

The condition $\mu_j < 0$ indicates that $\boldsymbol{\mu}$ under consideration is not physically possible. Thus, we argue that the value of h_i for $\ell < 0$ is somewhat arbitrary and it is not essential for it to match the usual log-likelihood function since negative values of ℓ are not physical. Therefore, we propose to replace the cost functions $h_i(\ell)$ for $\ell < 0$ with functions that are suited to our goal of controlling negative pixels.

The method to derive the modified likelihood is shown below. We divide the rays into three categories depending on their representative plots in Fig. 3.1 :

Category 1 (Highly attenuated rays) :	$I_1 = \{i : \check{y}_i \leq r_i\}$
Category 2 (Attenuated rays) :	$I_2 = \{i : r_i < \check{y}_i \leq r_i + b_i\}$
Category 3 (Almost unattenuated rays) :	$I_3 = \{i : r_i + b_i < \check{y}_i\}$

We propose the following modification to h_i. In categories 1 and 2, for $t < 0$ we replace h_i with a straight line such that the continuity of the function is maintained and the slope of the line is equal to $\dot{h}_i(0)$. A straight line is chosen as opposed to a parabola because it permits the surrogate to have a low curvature. Low curvatures are advantageous as they increase the convergence speed of the algorithm. For categories 1 and 2, we define :

$$\tilde{h}_i(t) \triangleq \begin{cases} h_i(t), & t \geq 0 \\ h_i(0) + \dot{h}_i(0)t, & t < 0. \end{cases}$$

In category 3, it is not possible to replace h_i for $t < 0$ with a straight line that has a negative slope. Having a line with negative slope would make h_i non-differentiable. We reject this choice as we only consider differentiable functions in this report. For sake of simplicity, we choose a parabola to replace h_i for $t < 0$. The parabola is chosen such that the continuity of h_i and $\dot{h}_i$ are maintained. For reasons of computational simplicity, the curvature of the parabola is computed using [1, eq. 29]. For category 3, we define :

$$\tilde{h}_i(t) \triangleq \begin{cases} h_i(t), & t \geq 0, \\ h_i(0) + \dot{h}_i(0)t + \frac{1}{2}\frac{(\check{y}_i - r_i)^2}{\check{y}_i}t^2, & t < 0. \end{cases}$$

The new negative log-likelihood can be thus written as follows :

$$(3.4) \quad -\tilde{L}(\boldsymbol{\mu}) = \sum_{i \in I_1} \tilde{h}_i([\boldsymbol{A}\boldsymbol{\mu}]_i) + \sum_{i \in I_2} \tilde{h}_i([\boldsymbol{A}\boldsymbol{\mu}]_i) + \sum_{i \in I_3} \tilde{h}_i([\boldsymbol{A}\boldsymbol{\mu}]_i),$$

$$(3.5) \quad \tilde{h}_i(t) \triangleq \begin{cases} h_i(t) & \text{if } t \geq 0, \quad i \in I_1 \cup I_2 \cup I_3, \\ h_i(0) + \dot{h}_i(0)t & \text{if } t < 0, \quad i \in I_1 \cup I_2, \\ h_i(0) + \dot{h}_i(0)t + \frac{1}{2}\frac{(y_i - r_i)^2}{y_i}t^2 & \text{if } t < 0, \quad i \in I_3. \end{cases}$$

A quadratic surrogate of $-\tilde{L}(\boldsymbol{\mu})$ was derived as :

$$\tilde{\phi}_L(\boldsymbol{\mu};\boldsymbol{\mu}^{(n,0)}) = \sum_{i=1}^{n_d} q_{Li}([\boldsymbol{A}\boldsymbol{\mu}]_i;[\boldsymbol{A}\boldsymbol{\mu}^{(n,0)}]_i), \tag{3.6}$$

$$q_{Li}(t;s) = \tilde{h}_i(s) + \dot{\tilde{h}}_i(s)(t-s) + \tfrac{1}{2}\breve{c}_i(s)(t-s)^2, \tag{3.7}$$

$$\breve{c}_i(s) = \begin{cases} \frac{\dot{h}_i(s)-\dot{h}_i(0)}{s}, & s>0, i \in I_1 \cup I_2, \\ \ddot{h}_i(0), & s \le 0, i \in I_1 \cup I_2, \\ \frac{(y_i-r_i)^2}{y_i}, & i \in I_3, \end{cases}$$

where, $\ddot{h}_i(t) = d^2 h_i(t)/dt^2$.

The proof of majorization of $\tilde{\phi}_L$ over $-\tilde{L}(\boldsymbol{\mu})$ on $S = \mathbb{R}^n_p$ is in Section A.2.2.

3.3 Simulations and results

The modified cost function was initially developed for transmission tomography, but the monoenergetic Poisson model is applicable to X-ray CT also. Transmission tomography is used to obtain attenuation maps for use in PET reconstruction in PET/CT systems. The simulations and results presented here are for transmission tomography. An algorithm that is well-known, but is missing from the picture here is LBFGS-B. LBFGS-B is a constrained algorithm for bound constrained minimization. It can be said to be roughly composed of an active constraint algorithm and an unconstrained algorithm. It is easily changed into an unconstrained algorithm by disabling all constraints (both in theory and computer code). LBFGS-B however has the drawback of storing more and more image sized volumes to increase the convergence rate. This memory-convergence rate trade-off of LBFGS-B will ultimately decide its use in practical situations.

The size of the attenuation map of the phantom used in the simulations was 128x128 and the size of each pixel was 4×4mm. The number of angles was 80 and number of bins per angle was 136. We used a simple forward and back projector based on simple geometrical interpolations to test the algorithms. A mask of support was computed around the phantom using the true phantom. The mask not only saves computation time by reducing the number of pixels to be estimated, it also incorporates the information that many pixels around the image represent air whose attenuation constant is close to zero. Poisson noise with parameters $b_i = 10^6$ and $r_i = 10$ was added to the sinogram. Initial estimate of the attenuation map, $\boldsymbol{\mu}_0$, was the FBP reconstruction with negative pixels to zero. The algorithms compared were

1. Constrained OS algorithm with 5 subsets and precomputed curvatures called OS-SPS-PC,
2. Unconstrained PCG algorithm with precomputed curvatures using PCG_LS_QS algorithm (Section 6.1.1) called QS-PCG-PC,
3. Unconstrained PCG algorithm with optimal curvatures using PCG_LS_QS algorithm (Section 6.1.1) called QS-PCG-OC, and
4. Unconstrained PCG algorithm on the modified cost function using QS_PCG_LS algorithm (Section 6.1.3) called QS-PCG-MOD.

Edge-preserving, space-invariant regularization based on the Huber potential function (see Table 2.1) was used. Its parameters, β and δ, were determined by trial-and-error to produce acceptable visual quality. The simulations were done using *MATLAB*.

The reconstructions from all the algorithms were similar (Fig. 3.2 and Fig. 3.3), and also had similar properties with respect to the negative pixels (Fig. 3.6). All the unconstrained algorithms attained similar values of cost functions (Fig. 3.4 and Fig. 3.5). It is also observed from these plots that the convergence rate of QS-PCG-MOD was slightly lower than the other algorithms. This is not a major disadvantage because a good agreement in cost function values is attained in a relatively small number of iterations.

3.4 Summary

The negative log-likelihood for the monoenergetic and Poisson statistics case was modified to permit the creation of a monotone, unconstrained PCG algorithm. This algorithm does not incur the 50% compute time overhead that was incurred with the requirement of monotonicity in previous PCG algorithms. The non-negativity constraint was not imposed explicitly and regularization was used to control the negative pixels. Since, the modification of the negative log-likelihood is similar to penalty based constrained optimization, negative pixels can be controlled further by increasing the penalty as the iterations proceed. Furthermore, the modification of the likelihood was done only for non-physical values of the image.

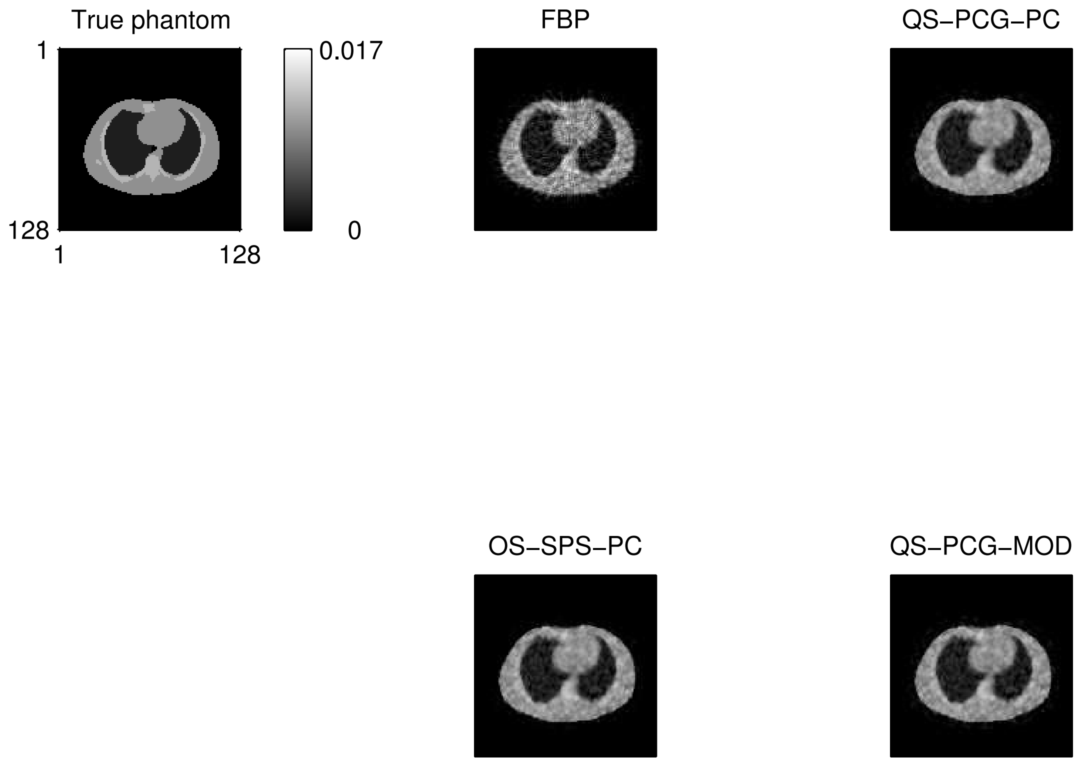


Figure 3.2: Images of the “True Phantom” and its reconstructions by various algorithms.

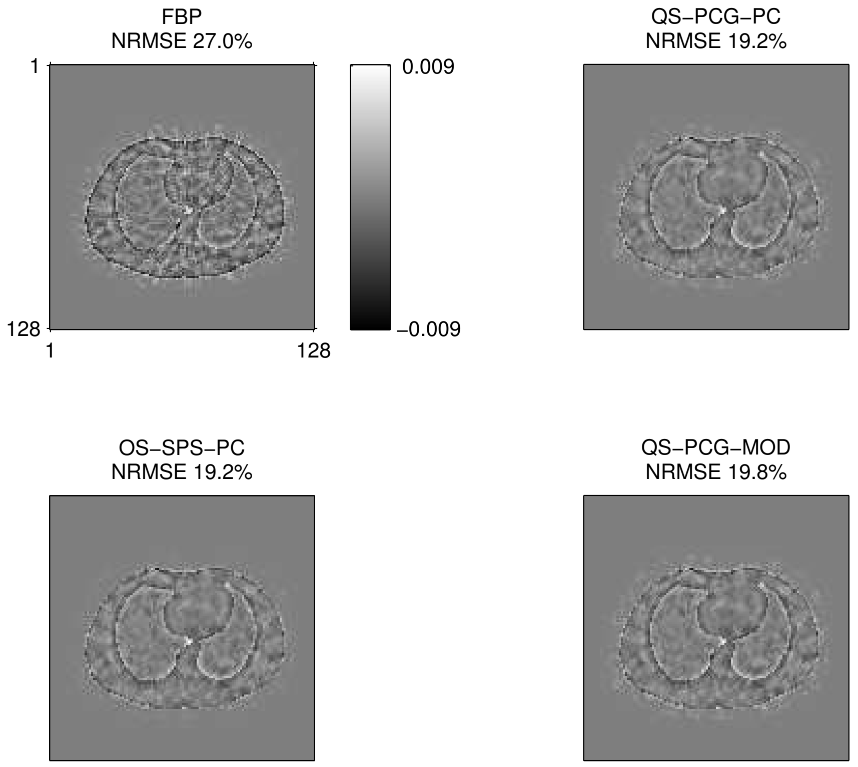


Figure 3.3: The difference images between the reconstructions and the true phantom. *NRMSE* is defined as $\|[\boldsymbol{\mu}^{(n)}]_+ - \boldsymbol{\mu}_{true}\|/\|\boldsymbol{\mu}_{true}\|$.

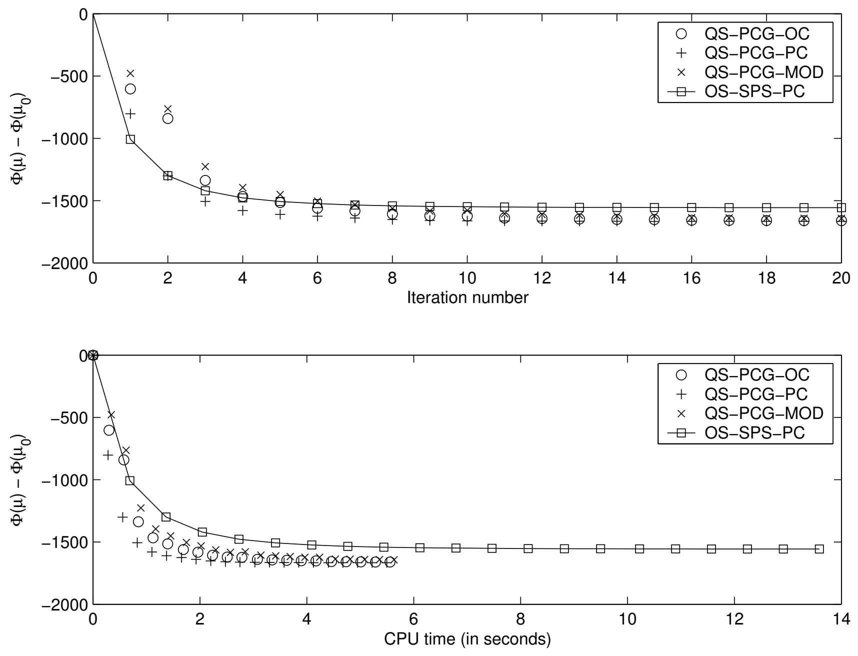


Figure 3.4: Plot showing the variation of $\Phi(\boldsymbol{\mu})$ with iteration number and CPU time.

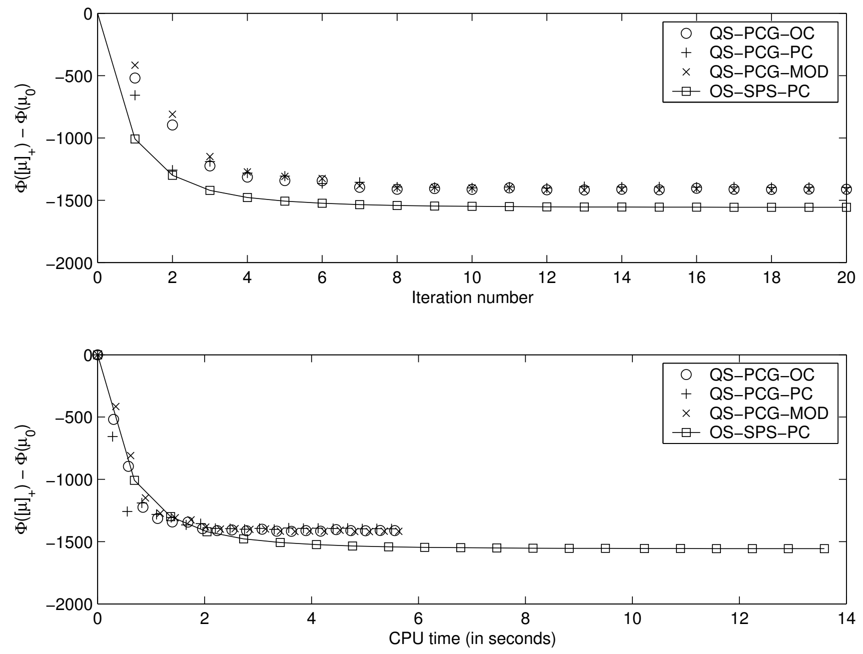


Figure 3.5: Plot showing the variation of $\Phi([\boldsymbol{\mu}]_+)$ with iteration number and CPU time.

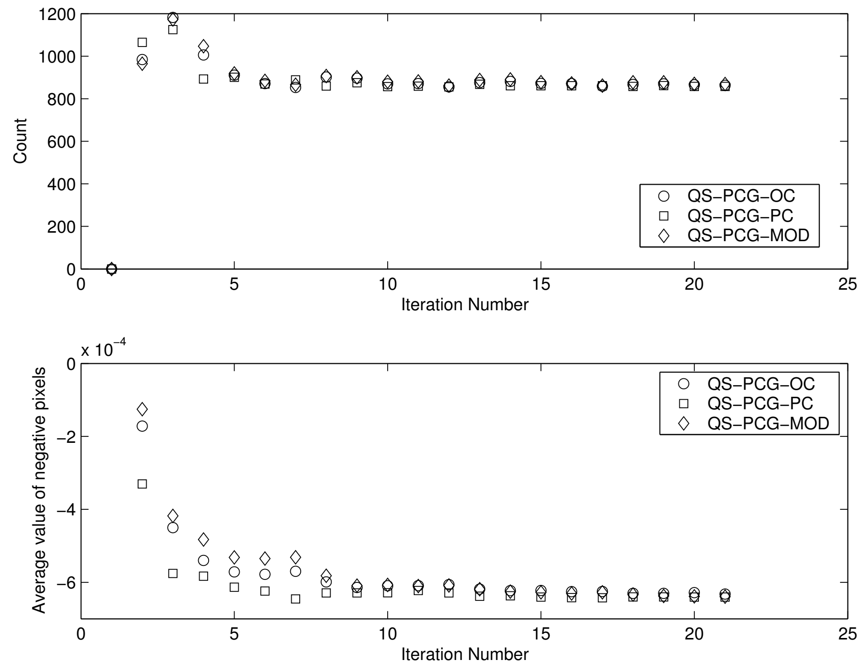


Figure 3.6: Plot showing the variation of number of negative pixels and the average value of negative pixels with iteration number.

CHAPTER 4

Simplified cost function for beam hardening

Iterative statistical methods that have been used to account for beam-hardening methods [14, 21] have used physical information that is not required by the JS method [38]. The disadvantage of the JS method is that it was devised for use with FBP reconstruction. Thus, it is desirable to adapt the JS method to the statistical image reconstruction philosophy. The simplified cost function developed here accomplishes this goal. The JS method is described in terms of our notation in Section 4.1. The adaptation of the JS method happens in a trivial fashion from that description in Section 4.2.

There are two ways of account for beam hardening in statistical methods. The first is to represent the X-ray attenuation coefficient using physical effects [14]. And the second is to represent the X-ray attenuation coefficient using basis materials [21]. Both methods have used more calibration information than is used by the JS method. In this section, we show how to modify the basis materials approach to use only the information used by the JS method to correct beam hardening artifacts.

4.1 Sinogram pre-correction for beam hardening : Joseph-Spital (JS) method

Artifacts due to beam-hardening are of two types : cupping and streaks. Cupping is observed in all uncorrected reconstructions, whereas streaks are present when bone content is high in the object. Cupping can be removed by water correction. Water correction is the process of computing the water-equivalent lengths for each ray. The water-equivalent lengths are defined as follows. First define, $f_S(T_S) \triangleq f(T_S, 0)$. Observe from Fig. 4.1(a) that f_S is an invertible function. It is called the water-correction table in [38]. $f_S^{-1}(l_i) \triangleq T_E^*$ is called the water-equivalent length. If the object consisted of soft-tissue only then T_E^* would be the exact line integral of soft tissue density and the cupping would be removed perfectly. Indeed, this has been found to be the case for abdominal scans [43]. The above approximation does not hold in head scans as the bone content is relatively higher.

Now, consider the two-material model of Section 2.5, and for simplicity, assume that the noise is absent. From (2.4), we would have [1] $f(T_{S,i}, T_{B,i}) = \ln(b_i/(y_i - r_i)) \triangleq l_i$. $f(T_S, T_B)$ is also referred to as the water and bone correction table. This set of equations can not be solved mathematically for $T_{S,i}$ and $T_{B,i}$ without invoking physical considerations and/or practical observations. This is because for every value of the function f, every point on a curve in the $(T_{S,i}, T_{B,i})$ plane is a solution (see Fig. 4.1(a)). The solutions for $T_{S,i}$ and $T_{B,i}$ are obtained by dual-energy methods and the JS method in two different ways. Dual-energy methods gather data at a second energy which provides a second curve in the $(T_{S,i}, T_{B,i})$ plane. Thus, $T_{S,i}$ and $T_{B,i}$ are solved for exactly. The JS method estimates the value of $T_{B,i}$ using a practical observation. The estimate of $T_{B,i}$ in turn gives us the value of $T_{S,i}$. Even though the JS method is not as rigorous as the Dual-energy methods, the artifact removal of the JS method has been found to be satisfactory. This gives the JS method an edge over the dual-energy methods as it uses one less scan and keeps the radiation dose low.

[1]There is a possibility of taking log of negative sinogram bins here in regions where y_i values are low *i.e.* rays passing though a lot of material. The extremely low values of y_i cause streaks. The value of y_i can be adjusted using an adaptive trimmed mean filter [30] to prevent noise streaks. A simpler solution could simply increase the offending values in order to provide non-negative line integrals. The second approach can cause bias and streaks in the reconstructions.

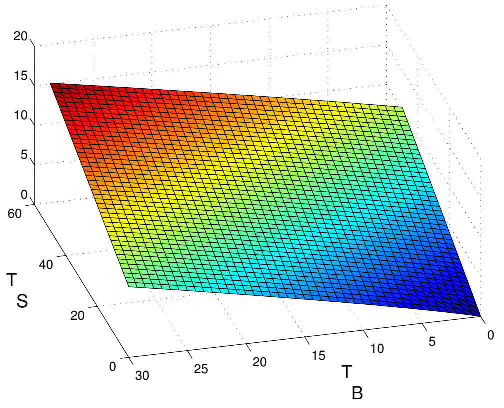
The water equivalent-lengths of bone regions are roughly in proportion to the density ratio of bone to soft-tissue. But, the effect is non-linear due to the fact that $m_S(\mathcal{E})$ and $m_B(\mathcal{E})$ are not related by a constant ratio at every energy [38]. To state the above fact in terms of f, the non-linear nature of f is responsible for beam-hardening artifacts [38]. Non-linearity of f increases with a decrease in voltage setting of the X-ray tube. $\gamma(T_S, T_B)$ is defined as the scaling required for the bone-tissue density length (T_B) to obtain the water-equivalent length *i.e.* it must satisfy : $f(T_S, T_B) = f(T_S + \gamma(T_S, T_B)T_B, 0)$. One can easily show that γ (see Fig. 4.1(b)) defined as follows, satisfies the above requirement :

$$\gamma(T_S, T_B) = \begin{cases} \frac{f_S^{-1}(f(T_S,T_B))-T_S}{T_B} & T_B \neq 0, \\ \lim_{T\to 0} \frac{f_S^{-1}(f(T_S,T))-T_S}{T} & T_B = 0. \end{cases}$$

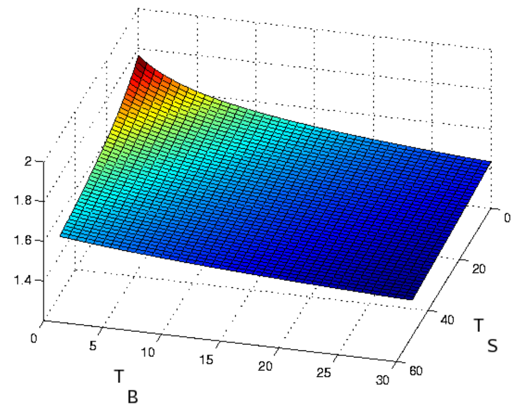
Joseph and Spital found that [2] $\gamma(T_S, T_B)$ is approximated adequately by $A - BT_B$, where A and B are free tuning parameters. This approximation is possible because in actual images the amount of bone is small enough and the viewing window for the image is large. A larger viewing window enables us to tolerate a larger amount of error. The JS method can be described in our notation as follows :

1. Compute $T_E^* = f_S^{-1}(l_i)$.
2. Back-project T_E^* to create a water-corrected reconstruction. Call it $\boldsymbol{\rho}_w$.
3. Segment the regions of $\boldsymbol{\rho}_w$ that exceed a given threshold ρ_1 as bone regions. Joseph and Spital found that the bone density is approximately over-estimated by a factor λ_0. Thus, forward project the estimated bone region and divide it by λ_0 to obtain T_B^*, the estimate of bone density integral.
4. Compute $T_E^{final} = T_S^* + T_B^*$ using $T_S^* + (A - BT_B^*)T_B^* = T_E^*$ and the values of T_E^* and T_B^* from steps 1 and 3 above.

[2] $\gamma(T_S, T_B) = \lambda_L$ in their notation.



(a) $f(T_S, T_B)$



(b) $\gamma(T_S, T_B)$

Figure 4.1: Plot of $f(T_S, T_B)$ and $\gamma(T_S, T_B)$. Units of T_S and T_B are g/cm^2.

5. $\check{l}_i = T_E^{final}$ is the sinogram to which FBP can be applied to get a beam-hardening-artifact free reconstruction. For statistical methods, $\check{y}_i = b_i \exp\left(-f_S(\check{l}_i)\right) + r_i$ is the quantity used in reconstructions.

4.2 Statistical reconstruction methods

The negative log-likelihood that takes beam-hardening into account was derived in (2.6). It is repeated here (after changing the definition of h_i) for convenience :

(4.1)
$$-L(\boldsymbol{\rho}) = \sum_{i=1}^{n_d} h_i(f([\boldsymbol{G}I_S\boldsymbol{\rho}]_i, [\boldsymbol{G}I_B\boldsymbol{\rho}]_i)),$$
$$h_i(t) \triangleq (b_i e^{-t} + r_i) - y_i \ln(b_i e^{-t} + r_i).$$

From Section 4.1 we have,

(4.2)
$$f(T_S, T_B) \approx f_S(T_S + (A - BT_B)T_B).$$

Combining the above equations gives us the likelihood for the statistical method that uses the JS method to account for beam-hardening. In the above equation f may be depend on the ray i when a bowtie-filter is used [62] and the cost function would be modified accordingly. Following [59], we have the following choices for statistical reconstruction methods :

1. *1-parameter method* : We use a constant to approximate $\gamma(T_S, T_B)$ instead of using (4.2), *i.e.*, set $B = 0$. This is equivalent to [38, Eq.(20)]. This method is used to verify that the amount of bone in the phantom is significant from the point of view of beam-hardening, and more than one tuning parameter is required for the approximation to be good.
2. *2-parameter method* : This leads to the simplified cost function and uses (4.2) to approximate $\gamma(T_S, T_B)$.
3. *Exact method* : The exact value of $\gamma(T_S, T_B)$, hence the exact value of $f(T_S, T_B)$, is known in simulations. This method is almost identical to [20].
4. *Ad hoc method* : In this method, we perform the statistical reconstruction assuming a water-correction model and process the obtained image by the JS method. We assume that the noise can be removed by including the water correction model (*i.e.* set $A = 1$ and $B = 0$) only in the measurement model. Now, the JS method is used to get rid of the beam-hardening artifacts. This is not a systematically derived method but is attractive due to its simplicity. It is included here for the purpose of comparison with the 2-parameter method.

4.2.1 1-parameter and Ad hoc methods

The negative log-likelihood for these methods are written as

$$-L(\boldsymbol{\rho}) = \sum_{i=1}^{n_d} h_i(f_S([G_1\boldsymbol{\rho}]_i)),$$

$$G_1 \triangleq \begin{cases} G, & \text{Ad hoc method,} \\ G(I_S + AI_B), & \text{1-parameter method.} \end{cases}$$

Using the optimal curvatures computed in [1], the approximate surrogate is expressed as

$$Q_1(\boldsymbol{\rho};\boldsymbol{\rho}^{(n)}) = \sum_{i=1}^{n_d} h_i(f_S([G_1\boldsymbol{\rho}^{(n)}]_i)) + h_i'(f_S([G_1\boldsymbol{\rho}^{(n)}]_i)) \times (f_S([G_1\boldsymbol{\rho}]_i) - f_S([G_1\boldsymbol{\rho}^{(n)}]_i)) + \frac{1}{2}\breve{c}_i(f_S([G_1\boldsymbol{\rho}]_i) - f_S([G_1\boldsymbol{\rho}^{(n)}]_i))^2.$$

Observations of $f_S(T_S)$ suggest that it is a concave function. Using Lemmas 3 and 4 of [1] we can prove the following :

$$f_S'(T_S^{(n)})(T_S - T_S^{(n)}) \geq f_S(T_S) - f_S(T_S^{(n)}),$$
$$\left|\frac{f_S(T_S^{(n)})}{T_S^{(n)}}\right| |T_S - T_S^{(n)}| \geq |f_S(T_S) - f_S(T_S^{(n)})|.$$

Observations of $f_S(T_S)$ also suggest that $f_S'(0) \geq |\frac{f_S(T_S^{(n)})}{T_S^{(n)}}|$. Thus, we have the final form for the surrogate :

$$Q_2(\boldsymbol{\rho};\boldsymbol{\rho}^{(n)}) = \sum_{i=1}^{n_d} h_i(f_S([G_1\boldsymbol{\rho}^{(n)}]_i)) + h_i'(f_S([G_1\boldsymbol{\rho}^{(n)}]_i)) \times f_S'([G_1\boldsymbol{\rho}^{(n)}]_i)[G_1(\boldsymbol{\rho} - \boldsymbol{\rho}^{(n)})]_i + \frac{1}{2}\breve{c}_i(f_S'(0))^2[G_1(\boldsymbol{\rho} - \boldsymbol{\rho}^{(n)})]_i^2.$$

The above surrogate for the log-likelihood is used in the QS_PCG_LS algorithm (Section 6.1.3). These methods will not have nice convergence properties like monotonicity due to the beam-hardening approximations made to the exact measurement model and the use of masks I_S and I_B. The 1-parameter method is initialized with a 1-parameter JS reconstruction and the Ad hoc method is initialized with a water-corrected FBP reconstruction. 1-parameter and Ad hoc methods require one forward projection and one back-projection every iteration.

4.2.2 2-parameter and Exact methods

The negative log-likelihood for these methods are written as :

$$-L(\boldsymbol{\rho}) = \sum_{i=1}^{n_d} h_i(f([GI_S\boldsymbol{\rho}]_i, [GI_B\boldsymbol{\rho}]_i)).$$

It might be possible to derive an exact 2-D surrogate for this negative log-likelihood along the lines of [1] but the procedure appears to be very complicated. A simpler way is to approximate $f(T_S, T_B)$ by a local Taylor series expansion for every pair of (T_S^n, T_B^n) at the current iterate and hope that this approximation holds for the step sizes generated :

$$f(T_S, T_B) \approx f(T_S^n, T_B^n) + \left[\begin{array}{cc} \dot{f}_{1,0}(T_S^n, T_B^n) & \dot{f}_{0,1}(T_S^n, T_B^n) \end{array} \right] \left[\begin{array}{c} T_S - T_S^n \\ T_B - T_B^n \end{array} \right].$$

The quadratic approximate surrogate can be now written as :

$$\begin{aligned} Q_3(\boldsymbol{\rho}; \boldsymbol{\rho}^{(n)}) &= K + ({G_2}^T \dot{\boldsymbol{h}})^T (\boldsymbol{\rho} - \boldsymbol{\rho}^{(n)}) + \frac{1}{2} (\boldsymbol{\rho} - \boldsymbol{\rho}^{(n)})^T {G_2}^T \text{diag}(\breve{c}_i) G_2 (\boldsymbol{\rho} - \boldsymbol{\rho}^{(n)}), \\ [\dot{\boldsymbol{h}}]_i &\triangleq \dot{h}_i(f([G I_S \rho^{(n)}]_i, [G I_B \rho^{(n)}]_i)), \\ G_2 &\triangleq \left[\begin{array}{cc} \text{diag}(\dot{f}_{1,0}) & \text{diag}(\dot{f}_{0,1}) \end{array} \right] \left[\begin{array}{cc} G & 0 \\ 0 & G \end{array} \right] \left[\begin{array}{c} I_S \\ I_B \end{array} \right]. \end{aligned}$$

The above surrogate for the log-likelihood is used in the QS_PCG_LS algorithm (Section 6.1.3). Due to the Taylor series expansion and use of masks, these algorithms are not guaranteed to be monotonic. However, they have been found to perform satisfactorily in practice. The 2-parameter method is initialized by a 2-parameter JS reconstruction and the Exact method is initialized by a JS method using the water and bone correction table. The 2-parameter and Exact methods require two forward and back projections every iteration.

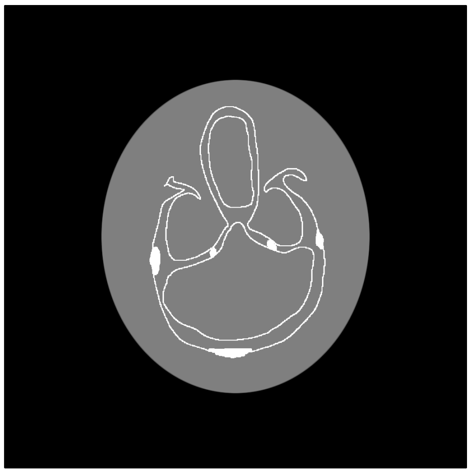
4.3 Simulation and results

A 2-D fan-beam X-ray CT scanner was simulated in this study. The geometry of the scanner is similar to the central slice of a GE Lightspeed Pro scanner. Sinogram data was collected over a 360° rotation over a 50cm field-of-view (FOV). The sinogram dimensions were 984 angles by 888 bins per angle. An 80kVp spectrum was used and the blank scan count summed over the entire X-ray spectrum was 1.1×10^6 per bin. r_i were set to zero for this study. The phantom size was 1024×1024 pixels and the reconstruction was done on a 256×256 grid. A 1024 size grid permits the phantom to have an average bone mineral width of about 1.5mm.

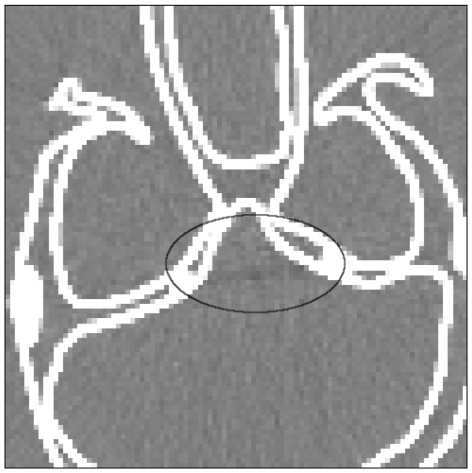
The reconstruction from the water-corrected FBP method (Fig. 4.2) is noisy and contains a beam hardening streak artifact in the center of the image. The 1-parameter method (Fig. 4.3 and Fig. 4.4, top left) provides a reconstruction which is free from noise but it still contains the streak. The reconstruction from the 2-parameter method (Fig. 4.3 and Fig. 4.4, top right) is free of the noise as well as the streak artifact. The image quality is comparable to that of the Exact method (Fig. 4.3 and Fig. 4.4, bottom left). The Ad hoc method (Fig. 4.3 and Fig. 4.4, bottom right) removes the beam hardening streak artifact but fails to remove the noise completely. The 2-parameter method should be the method of choice when compared to the Ad hoc method as it follows the measurement model more closely and uses the same beam-hardening information (water-correction table and tuning parameters).

4.4 Summary

Polyenergetic statistical image reconstruction methods developed in the past used more beam-hardening calibration information than the FBP methods. The FBP methods achieved significant artifact reduction by using the sinogram pre-correction technique known as the JS method. The polyenergetic statistical method developed here eliminates the need to obtain the extra beam-hardening calibration information, and reuses the parameters used in the JS method.



(a) True phantom



(b) JS water-corrected FBP reconstruction

Figure 4.2: Image of the true phantom and JS water-corrected FBP reconstruction. Window = 400HU. Note the beam-hardening streak in the center of the FBP reconstruction.

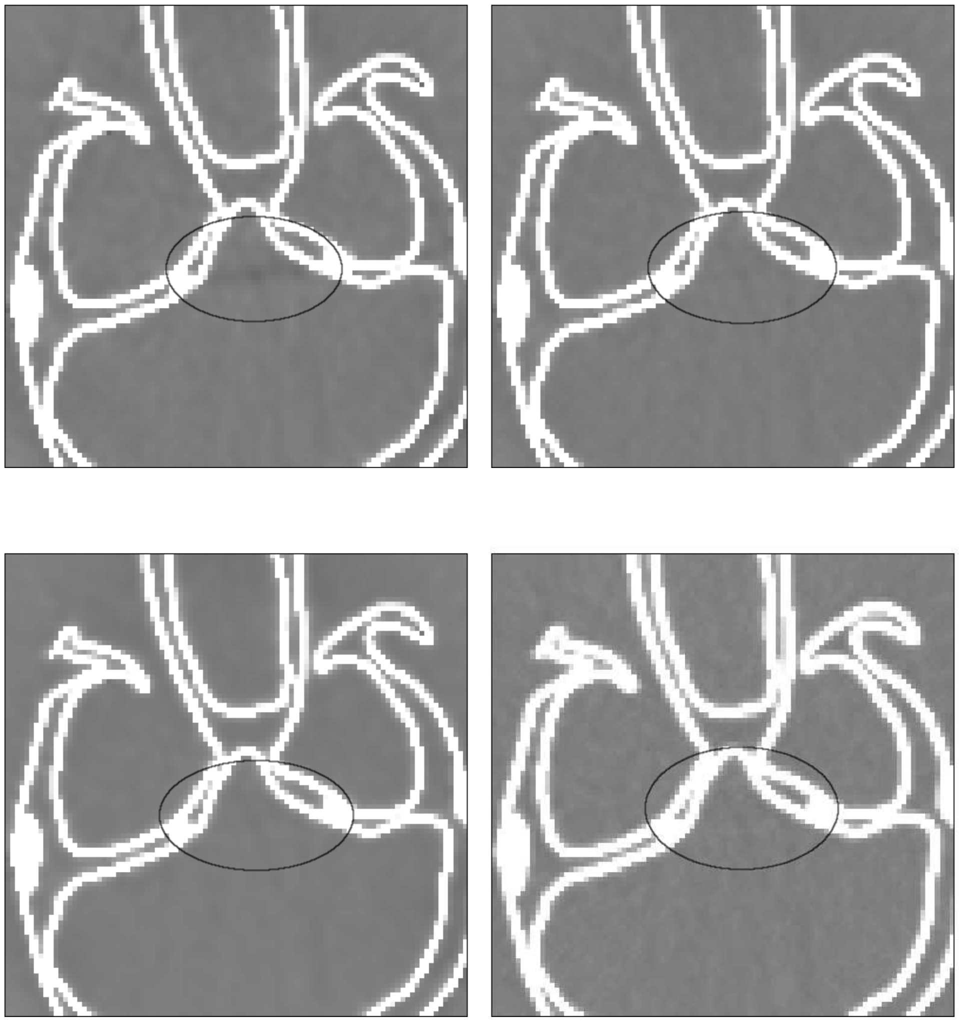


Figure 4.3: Reconstruction using statistical methods. Window = 400HU. *Clockwise from top left :* 1-parameter, 2-parameter, Ad hoc, Exact.

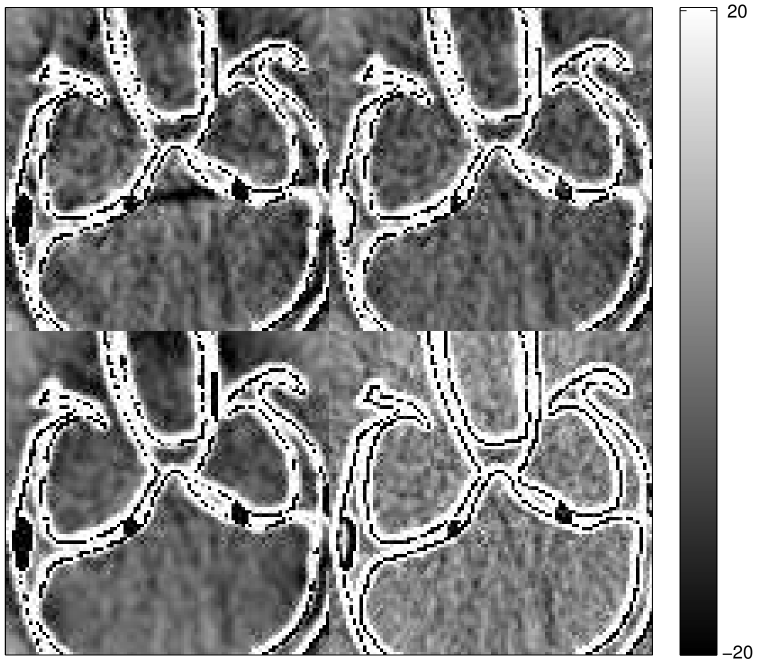


Figure 4.4: Difference images between reconstructions using statistical methods and the true phantom. *Clockwise from top left :* 1-parameter, 2-parameter, Ad hoc, Exact.

CHAPTER 5

Statistical method to reduce organ motion artifacts

This is a short chapter that investigates a solution to a particular, simulated, simple, organ motion problem in X-ray CT. The goal here is to create a motion-compensated statistical reconstruction method by using only the observed projections and simple image-processing operations like thresholding. We hope that the perspectives about motion developed in the given solution to this simple problem, will be applicable to more practical problems[1]. Section 5.1 presents background on the motion problem in general, and X-ray CT in particular. Section 5.2 defines the simulated problem being investigated here. Section 5.3 presents the proposed solution to the problem defined in Section 5.2. Finally, Section 5.5 summarizes the proposed solution, its advantages, and its disadvantages.

5.1 Background

The problem of motion is a very basic one. When a patient is in the scanner, there is motion due to a lot of phenomena: breathing, beating heart, peristalsis (*e.g.*, motion of esophagus and intestine), muscle contractions and expansions, movement due to discomfort *etc.*. Therefore, the single image produced by the scan is some sort of an average over the duration of the scan. Some kinds of motion, like breathing and cardiac motion are

[1]Special thanks to Bruno De Man, for sharing his intuition on this topic.

quasi-periodic, while in other kinds, the moving objects do not return to their respective initial states during the scan. We focus on the second problem, even though both kinds of motion are closely related. An example of the type of problem that we wish to focus on is as follows. In a lung scan whose duration is much less than the duration of a single breath, the diaphragm wall would move a certain distance during the scan and not return to its initial state.

To image quasi-periodic motion, a plethora of scanning techniques and image reconstruction methods have been developed in the past few years. Even a new scanner with two X-ray sources has been proposed. An example of image reconstruction with quasi-periodic motion is described next. In cardiac X-ray CT, an image of the heart is required to be produced at a particular phase in the electro-cardiogram (ECG) signal. An ECG signal is collected along with projection data. In some cases, projection-data collection is triggered by the ECG signal, called prospective-gating, and in others, projection data is collected for a duration of time and the relevant pieces of projection data are selected based on the ECG signal, called retrospective gating. Thus, for cardiac X-ray CT, it is possible that we may not be able to collect all the required data for a given heart-phase. Iterative algorithms produce images with lower artifact than FBP based methods when such a data-insufficiency is encountered [49]. In [33], authors have proposed an new scanning geometry that reduces the X-ray dose required for cardiac imaging.

One way to beat the motion problem is to take all the required observations in a time that is much shorter than the motion. This kind of solution is attempted with a electron-beam computed tomography (EBCT) scanner. It has no moving parts and the X-ray beam is rotated electronically. One rotation takes only 50 milliseconds, which is much smaller than the duration of a heart beat at 120 beats per minute, 0.5 seconds. EBCT scanners however are very expensive and not easily available. So, newer scanning techniques and/or image reconstruction methods for the slower gantry based MSCT scanners would have to be developed. A solution to the motion problem has been tested in [17] by adding another x-ray source to the scanner similar to a flying focal spot, which the authors state is very costly. In an accompanying paper [18], the same authors propose a signal processing approach that is slightly sub-optimal but removes the need for an extra X-ray source. The suggested methods are hard to adapt to more general problems.

Previous work related to the current problem has been explored with FBP based methods. In [12], the authors created a parametric motion model to reduce motion artifacts caused by breathing. Some solutions to similar problems have been proposed in related fields. In cardiac X-ray CT, projections have been used to synthesize an signal called the kymogram that is an analogue of the ECG signal [39]. In single photon emission computerized tomography (SPECT), the authors of [50] compute an optical flow vector field from the observed tomographic views.

The approach on which the motion-compensated image reconstruction method proposed here is derived is different from the currently known methods. The current approach does not parameterize the object being imaged with a time variable. Neither does it attempt to difference consecutive views to obtain motion information. The proposed method first creates a motion-uncompensated reconstruction by traditional methods. And then it uses this reconstruction and the observed sinogram to create a motion-compensated reconstruction.

5.2 Problem definition

A simulation with a single-slice, fan-beam scanner is setup as follows. An object is scanned with a 360 degree rotation employing 984 views. The object is a modified Shepp-logan phantom which remains in state 1 (Fig. 5.1(left)) for views 1 to 491, and in state 2

(Fig. 5.1(right)) for views 492 to 984. One can notice that the state transition occurs due to a moving ellipse located in the top-center of the object. At view 492, the ray from the X-ray source to the center of detector array is roughly parallel to the major axis of the moving ellipse. No noise is added to the observed sinogram.

5.3 Proposed statistical reconstruction method

The proposed method is based on our intuitive understanding of the X-ray CT scanner. It is a pre- and post-processing type of a method, and we do not claim it has any optimality properties. Also, the proposed method is tailored to the problem definition of Section 5.2. However, it does provide some new insights into object motion in X-ray CT scans. The steps in the proposed method are as follows :

1. A regular (*i.e.* motion-uncompensated) reconstruction is made using FBP (a regularized iterative algorithm can be used when observations are noisy).
2. The difference sinogram between the forward projection of the motion uncompensated reconstruction and the observed sinogram is called the residual sinogram here. In the residual sinogram, bins outside the range $[-1000, 1000]$ HU are set to 1 and the rest to 0 to produce a second sinogram (Fig. 5.2(top-left)). This sinogram tells us the observations that are going to contain a large residual. The moving object would be mainly responsible for the large residuals because the the initial reconstruction did not take motion into account.
3. The above sinogram is back-projected (Fig. 5.2(top-right)). This image roughly tells how much various regions of the object contribute to the the large-residual regions of the residual sinogram.
4. The above image is thresholded at 300 to produce the image mask shown in Fig. 5.2(bottom-left). This mask, called motion mask, roughly tells us where the motion could be

happening. In other words, within the motion mask, image pixels change with view number, and outside the motion mask they do not.

5. The method to compute the view at which object motion happens is as follows. In the residual sinogram, we consider only those rays that pass through the motion mask, and set the rest to 0 (Fig. 5.3(left)). Now, the first half of the views of the observed sinogram come from an object in which the moving ellipse was large in size, and the second half comes from an object in which the ellipse was smaller. But, the motion-uncompensated reconstruction takes into account all views, producing a reconstruction that is intermediate between first and second states of the object (Fig. 5.4(left)). So, the observed sinogram would be greater than the forward projection of the motion-uncompensated sinogram in the first half of the views, and smaller in the second half. This change in sign could roughly tell us the view where the object moves from state 1 to state 2. Thus, we sum the residual sinogram over the detector dimension, and produce a one dimensional function (Fig. 5.3(right)). The sign change in this function gives us an estimate of the view at which object motion happens. The transition view, called $i^{\{1,2\}}$, is computed here as 491.

   Note here that this ad hoc procedure of estimating $i^{\{1,2\}}$ should work in the presence of noise also, due to the averaging taking place over the sinogram detector dimension.

6. During reconstruction, pixels outside the motion mask are reconstructed using all views. But, pixels within the motion mask are changing with view number. To account for this fact, multiple images are reconstructed within the motion mask as described below. A mapping between these images and view numbers is also created. In the current problem, there are 2 images within the motion mask: the first image corresponds to views 1 to $i^{\{1,2\}}$ and the second image to views $i^{\{1,2\}}$ to 984.

7. The negative log-likelihood for the weighted-least-squares and motion-uncompensated

case is

$$-L(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G}\boldsymbol{\mu}]_i - l_i)^2,$$

where, $\boldsymbol{\mu}$ is the image being estimated, n_d is the number of observations (*i.e.* sinogram bins), l_i and w_i are the line-integral observation and weight at sinogram bin i, and $\boldsymbol{G}$ is the forward projection operator. It is modified for motion as follows :

$$\begin{aligned}
&-L_{motion}(\boldsymbol{\mu}_{\mathcal{I}'_m}, \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m}, \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m}) \\
&= \sum_{i=1}^{i^{\{1,2\}}} \tfrac{1}{2} w_i([\boldsymbol{G}\boldsymbol{\mu}^{\{1\}}]_i - l_i)^2 + \sum_{i=i^{\{1,2\}}+1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G}\boldsymbol{\mu}^{\{2\}}]_i - l_i)^2, \\
&\qquad (= -L(\boldsymbol{\mu}),\ \text{for } \boldsymbol{\mu}^{\{1\}} = \boldsymbol{\mu}^{\{2\}} = \boldsymbol{\mu}) \\
&= \sum_{i=1}^{i^{\{1,2\}}} \tfrac{1}{2} w_i([\boldsymbol{G} \begin{bmatrix} \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2 + \sum_{i=i^{\{1,2\}}+1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G} \begin{bmatrix} \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2, \\
&= \sum_{i=1}^{i^{\{1,2\}}} \tfrac{1}{2} w_i([\boldsymbol{G} \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2 + \sum_{i=i^{\{1,2\}}+1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G} \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2, \\
&= \sum_{i=1}^{i^{\{1,2\}}} \tfrac{1}{2} w_i([\left[\begin{array}{c|c} \boldsymbol{G}_{\mathcal{I}'_m} & \begin{matrix} \boldsymbol{G}^{\{1\}}_{\mathcal{I}_m} \\ \\ \boldsymbol{G}^{\{2\}}_{\mathcal{I}_m} \end{matrix} \end{array}\right] \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2 \\
&\qquad + \sum_{i=i^{\{1,2\}}+1}^{n_d} \tfrac{1}{2} w_i([\left[\begin{array}{c|c} \boldsymbol{G}_{\mathcal{I}'_m} & \begin{matrix} \boldsymbol{G}^{\{1\}}_{\mathcal{I}_m} \\ \\ \boldsymbol{G}^{\{2\}}_{\mathcal{I}_m} \end{matrix} \end{array}\right] \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m} \end{bmatrix}]_i - l_i)^2, \\
&= \sum_{i=1}^{i^{\{1,2\}}} \tfrac{1}{2} w_i([\boldsymbol{G}_{\mathcal{I}'_m} \boldsymbol{\mu}_{\mathcal{I}'_m}]_i + [\boldsymbol{G}^{\{1\}}_{\mathcal{I}_m} \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m})]_i - l_i)^2 \\
&\qquad + \sum_{i=i^{\{1,2\}}+1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G}_{\mathcal{I}'_m} \boldsymbol{\mu}_{\mathcal{I}'_m}]_i + [\boldsymbol{G}^{\{2\}}_{\mathcal{I}_m} \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m}]_{i - i^{\{1,2\}}} - l_i)^2.
\end{aligned}$$

The definitions of various symbols are as follows.

- $\boldsymbol{\mu}^{\{1\}}$ and $\boldsymbol{\mu}^{\{2\}}$: Full-size images for object states 1 and 2 respectively.
- $i^{\{1,2\}}$: The sinogram bin at which state transition is estimated to occur.
- $\mathcal{I}_m$ and $\mathcal{I}'_m$: Motion mask and its complement respectively.
- $\boldsymbol{\mu}^{\{1\}}_{\mathcal{I}'_m}$ and $\boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m}$: Regions of $\boldsymbol{\mu}^{\{1\}}$ outside and inside the motion mask respectively.
- $\boldsymbol{\mu}^{\{2\}}_{\mathcal{I}'_m}$ and $\boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m}$: Regions of $\boldsymbol{\mu}^{\{2\}}$ outside and inside the motion mask respectively.
- $\boldsymbol{\mu}_{\mathcal{I}'_m}$: Outside the motion mask, the same parameters are to be estimated. Therefore, $\boldsymbol{\mu}_{\mathcal{I}'_m} \triangleq \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}'_m} \triangleq \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}'_m}$.
- $\boldsymbol{G}_{\mathcal{I}'_m}$: Forward-projector that projects image regions outside the motion mask to all sinogram bins.
- $\boldsymbol{G}^{\{1\}}_{\mathcal{I}_m}$: Forward-projector that projects image regions inside the motion mask to sinogram bins 1 to $i^{\{1,2\}}$.
- $\boldsymbol{G}^{\{2\}}_{\mathcal{I}_m}$: Forward-projector that projects image regions inside the motion mask to sinogram bins $i^{\{1,2\}}+1$ to n_d.

8. Suitable image regularization terms are added to $-L_{motion}(\boldsymbol{\mu}_{\mathcal{I}'_m}, \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m}, \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m})$ and alternating minimization is performed to estimate $\boldsymbol{\mu}_{\mathcal{I}'_m}$, $\boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m}$ and $\boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m}$. Finally, the above estimates are combined to obtain the final reconstructed images for motion states 1 and 2 : $\boldsymbol{\mu}^{\{1\}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{1\}}_{\mathcal{I}_m} \end{bmatrix}$ and $\boldsymbol{\mu}^{\{2\}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}'_m} \\ \boldsymbol{\mu}^{\{2\}}_{\mathcal{I}_m} \end{bmatrix}$.

5.4 Results

The proposed image reconstruction method assumes that the object being scanned has two states and a transition view at which the state transition occurs. For the problem setup here, this is indeed the case. The proposed method finds the transition view correctly as 491 as shown in Fig. 5.3(right). The proposed method is also able to localize correctly the image region where the motion occurs, *i.e.*, the motion mask is accurate in Fig. 5.4(right). The reconstruction produced by the proposed method has lower motion artifact than that produced by the traditional method. This fact is evident when we visually compare Fig. 5.4 and Fig. 5.5. More evidence for this can be seen in Fig. 5.6.

5.5 Summary

The solution proposed here has been tested in the absence of noise and works satisfactorily for the fan-beam case. In the presence of noise, two parameters estimated by image-processing operations, namely the motion mask and the transition view, will have to be adjusted as the iterations proceed. The proposed solution would only provide an initializer to these parameters. One problem is that, the image parameterization changes with the estimate of the motion mask. The dependence of the image parameters on the observed data also changes with the estimate of the transition view. The motion mask could expand to cover the entire image if it is not regularized properly. Because of these facts, the properties of the negative log-likelihood can not be predicted in the presence of noise. More investigation is required.

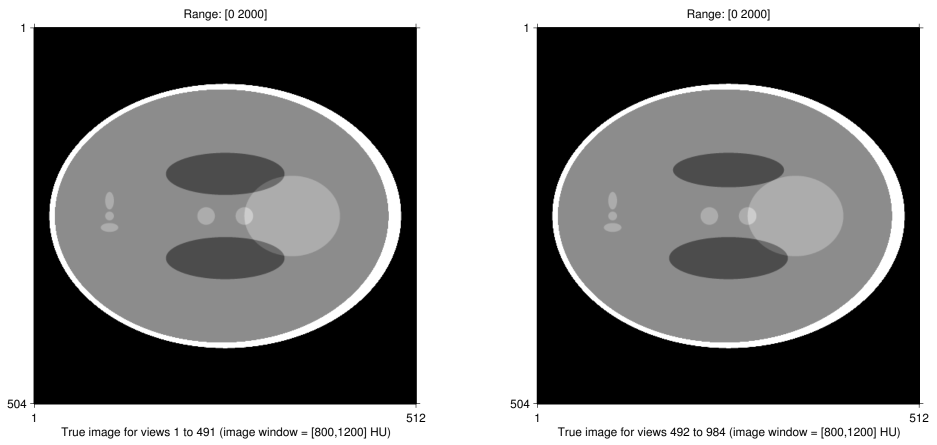

Range: [0 2000]
1
504
1
512
True image for views 1 to 491 (image window = [800,1200] HU)
Range: [0 2000]
1
504
1
512
True image for views 492 to 984 (image window = [800,1200] HU)

Figure 5.1: True images. *Left:* Object in state 1, and *Right:* Object in state 2. Image window = $[800, 1200]$ HU.

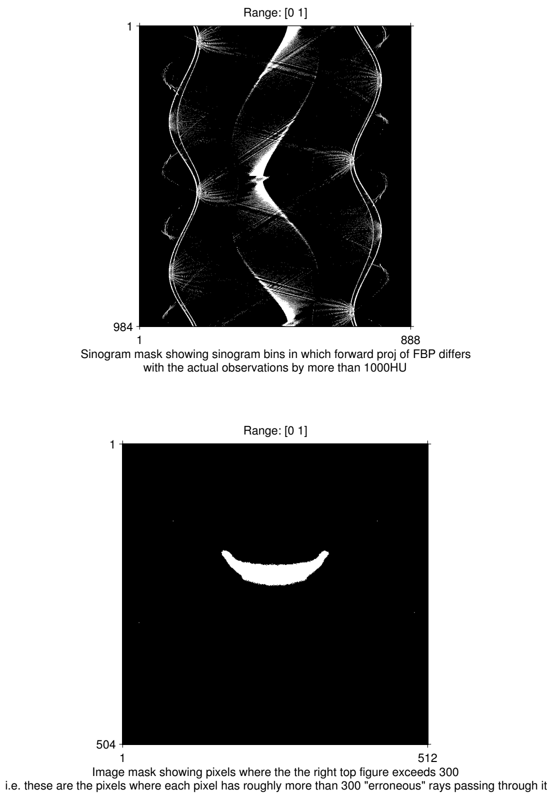

Range: [0 1]
1
984
1
888
Sinogram mask showing sinogram bins in which forward proj of FBP differs with the actual observations by more than 1000HU
Range: [0 564.212]
1
504
1
512
Back−projection of the sinogram in the left top figure
Range: [0 1]
1
504
1
512
Image mask showing pixels where the the right top figure exceeds 300
i.e. these are the pixels where each pixel has roughly more than 300 "erroneous" rays passing through it

Figure 5.2: Motion mask computation procedure. *Top-left:* Thresholded residual sinogram, *Top-right:* Back projection of thresholded residual sinogram, and *Bottom-left:* Motion mask.

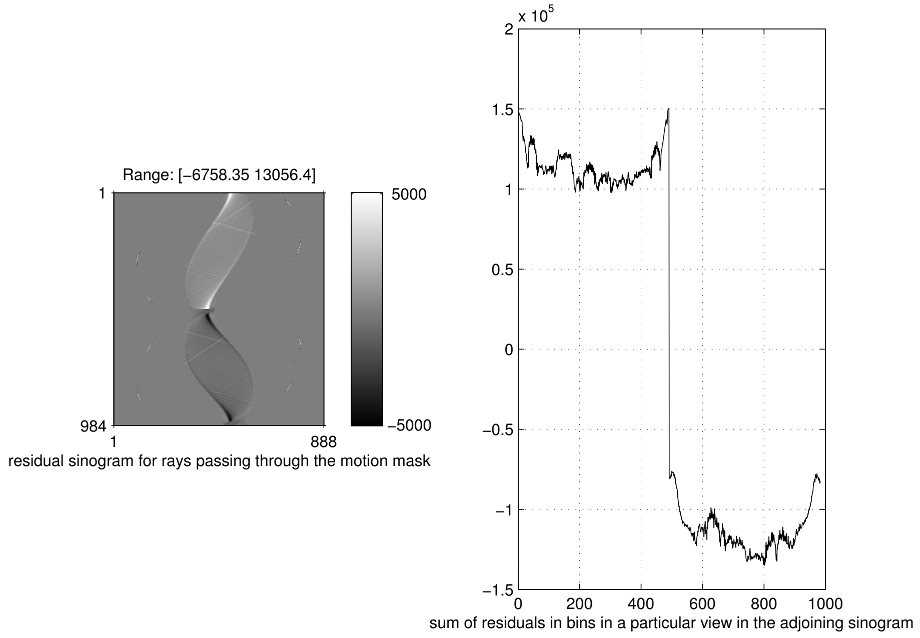


Figure 5.3: Computing the transition view. *Left:* Residual sinogram for rays passing through motion mask, and *Right:* Sum of sinogram on the left over the detector dimension.

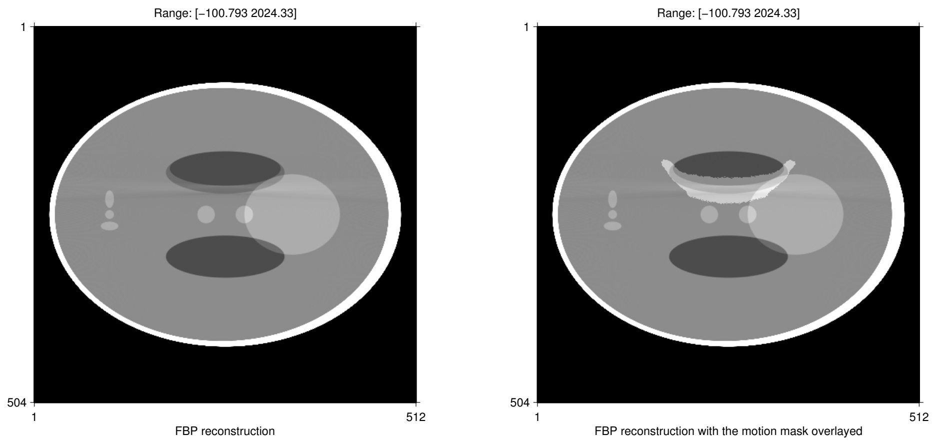

Range: [−100.793 2024.33]
1
504
1
512
FBP reconstruction
Range: [−100.793 2024.33]
1
504
1
512
FBP reconstruction with the motion mask overlayed

Figure 5.4: FBP reconstruction and motion mask. *Left:* FBP reconstruction, and *Right:* FBP reconstruction with motion mask superimposed. Image window = $[800, 1200]$ HU.

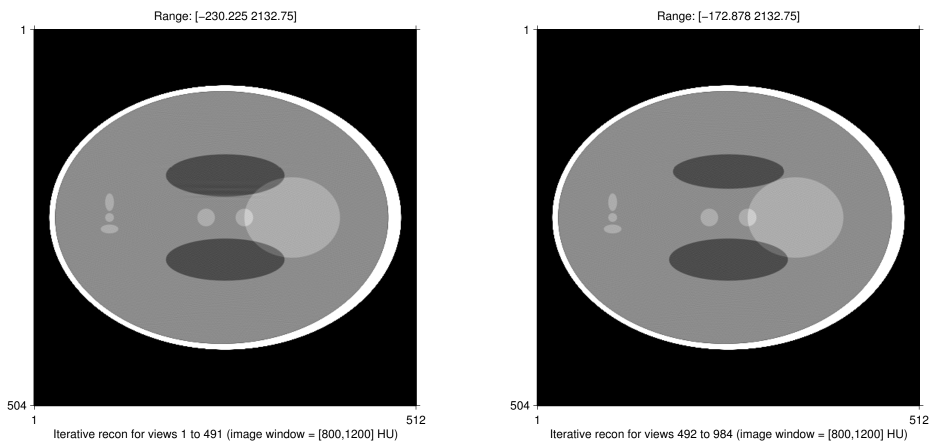

Range: [−230.225 2132.75]
1
504
1
512
Iterative recon for views 1 to 491 (image window = [800,1200] HU)
Range: [−172.878 2132.75]
1
504
1
512
Iterative recon for views 492 to 984 (image window = [800,1200] HU)

Figure 5.5: Iterative reconstruction. *Left:* Motion state 1, and *Right:* Motion state 2. Image window = $[800, 1200]$ HU.

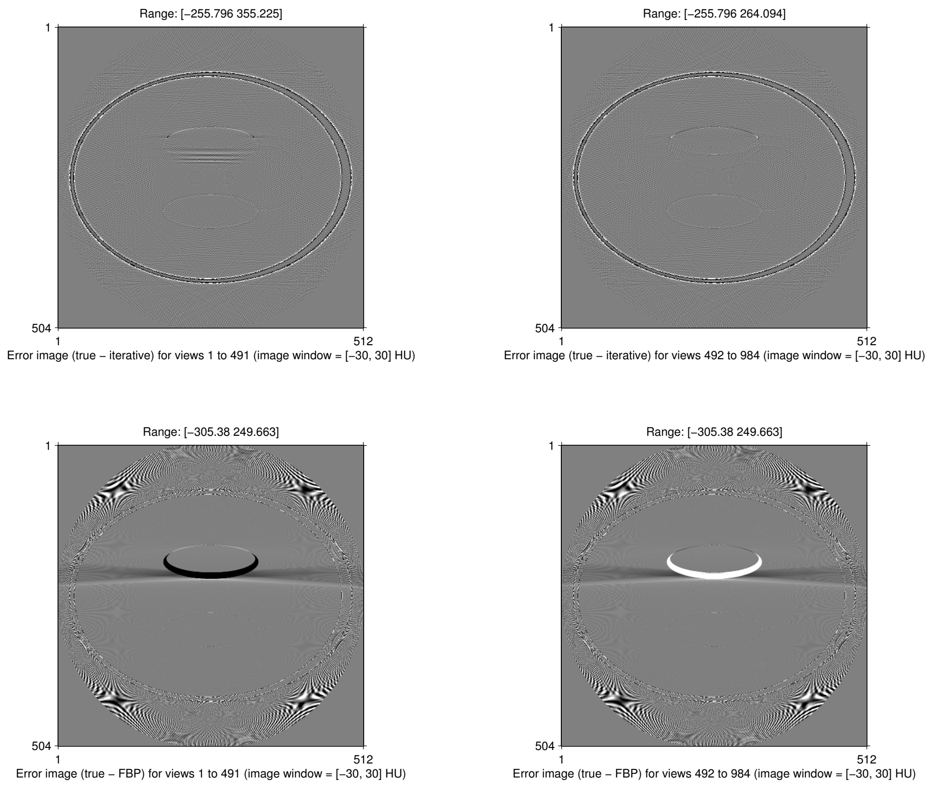

Range: [−255.796 355.225]
1
504
1
512
Error image (true − iterative) for views 1 to 491 (image window = [−30, 30] HU)
Range: [−255.796 264.094]
1
504
1
512
Error image (true − iterative) for views 492 to 984 (image window = [−30, 30] HU)
Range: [−305.38 249.663]
1
504
1
512
Error image (true − FBP) for views 1 to 491 (image window = [−30, 30] HU)
Range: [−305.38 249.663]
1
504
1
512
Error image (true − FBP) for views 492 to 984 (image window = [−30, 30] HU)

Figure 5.6: Error images. *Top-left:* True motion state 1 - iterative motion state 1, *Top-right:* True motion state 2 - iterative motion state 2, *Bottom-left:* True motion state 1 - FBP, and *Bottom-right:* True motion state 2 - FBP. Image window = $[-30, 30]$ HU.

CHAPTER 6

Comparisons between OS and PCG algorithms

This chapter compares OS-based and PCG-based algorithms. Section 6.1 compares two PCG algorithms: PCG_LS_QS (or, PLQ, for short) and QS_PCG_LS (or, QPL, for short). PCG_LS_QS has been known in literature [25] but is not guaranteed to minimize non-quadratic functions. So, we derive a new PCG algorithm, QS_PCG_LS, that is guaranteed to do so. This guarantee however comes at the price of slightly reduced convergence rate. Section 6.2 describes the Ordered Subsets (OS) idea. OS algorithms for image reconstruction in X-ray CT and other closely related fields have been known [2, 8, 35], and we summarize the PWLS_OS_SPS algorithm in this section. In Section 6.3, we compare PWLS_OS_SPS and PWLS_PLQ algorithms and show that PWLS_OS_SPS converges faster when the iterates are far away from the minimum/minima. But, we also remark that the potential of PCG algorithms is still untapped because preconditioners developed in other fields have not yet been used in X-ray CT.

6.1 Conjugate-gradient algorithms

6.1.1 PCG_LS_QS (or PLQ)

The PCG_LS_QS algorithm computes the descent direction from the original cost function using the PCG idea of (2.12). We note here that the descent in this direction on the original cost function is a line search on the original cost function (2.13). This line search is equivalent to the minimization of a single-variable non-quadratic function. In the original cost function, the notation of the negative log-likelihood and penalty function are combined for the purposes of this algorithm as follows :

$$\Phi(\boldsymbol{\mu}) \triangleq -L(\boldsymbol{\mu}) + R(\boldsymbol{\mu}) \triangleq \sum_{i=1}^{n'_d} h'_i([\boldsymbol{A}\boldsymbol{\mu}]_i).$$

At the current iterate $\boldsymbol{\mu}^{(n)}$, the search direction $\boldsymbol{d}^{(n)}$ is computed by using PCG (2.12) on $\Phi(\boldsymbol{\mu})$. Once the descent direction is computed, the single-variable non-quadratic cost function is defined as :

$$\Phi'(\alpha) \triangleq \sum_{i=1}^{n'_d} h'_i(\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha \boldsymbol{d}^{(n)})).$$

To minimize $\Phi'(\alpha)$, we design quadratic surrogates with respect to α for each of N_2 subiterations:

$$\begin{aligned}\Phi'(\alpha) \leq \phi'(\alpha; \alpha_m) &\triangleq \sum_{i=1}^{n'_d} q'_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha \boldsymbol{d}^{(n)})]_i; [\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha_m \boldsymbol{d}^{(n)})]_i), \quad \text{(QS using (2.11))} \\ &= \sum_{i=1}^{n'_d} \dot{h}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i(\alpha - \alpha_m) \\ &\quad + \tfrac{1}{2}\breve{c}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i^2(\alpha - \alpha_m)^2.\end{aligned}$$

Since $\phi'(\alpha; \alpha_m)$ is quadratic in α, the next iterate of α is obtained analytically as follows:

$$\begin{aligned}\alpha^{m+1} &= \text{arg min } \phi'(\alpha; \alpha_m), \\ \Rightarrow \alpha^{m+1} &= \alpha^m - \frac{\sum_{i=1}^{n'_d} \dot{h}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i}{\sum_{i=1}^{n'_d} \breve{c}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)} + \alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i^2}.\end{aligned} \tag{6.1}$$

The pseudo code is

```
1.        Initialize $\boldsymbol{\mu}^{(0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.          for n=0,1,2,...,$N_1$-1
2.1             Compute $\boldsymbol{d}^{(n)}$ using (2.12) and initialize $\alpha_m = 0$
2.2             for m=0,1,2,...,$N_2$-1
2.2.1                $\alpha_{m+1} = \alpha^m - \frac{\sum_{i=1}^{n'_d} \dot{h}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)}+\alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i}{\sum_{i=1}^{n'_d} \breve{c}_i([\boldsymbol{A}(\boldsymbol{\mu}^{(n)}+\alpha_m \boldsymbol{d}^{(n)})]_i)[\boldsymbol{A}\boldsymbol{d}^{(n)}]_i^2}$
2.2′            end
2.3             $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \alpha_{N_2} \boldsymbol{d}^{(n)}$
2′          end
```

The total number of forward and back projection operations in this algorithm is $2 \times N_1$.

This algorithm is guaranteed to be monotonic; the cost function is reduced every iteration. For non-quadratic cost functions, this algorithm could get stuck in unfavorable directions as a surrogate that majorizes the cost function globally is not created. Even though for a quadratic cost function, this algorithm is guaranteed to find a solution in a finite number of steps, quadratic cost functions are rarely useful in image reconstruction. Quadratic cost functions do not allow us to perform edge-preserving image reconstructions, usually yielding either too blurry (high β) or too noisy (low β) reconstructions. To guarantee the minimization of non-quadratic cost functions, the algorithm QS_PCG_LS (Section 6.1.3) should be used.

6.1.2 PWLS_PCG_LS_QS (or PWLS_PLQ)

The PWLS_PLQ algorithm is obtained by setting the likelihood part in the previous section for WLS using (2.5). The chosen penalty function and β are substituted in the regularization part to obtain the final PWLS_PLQ algorithm.

6.1.3 QS_PCG_LS (or QPL)

The QS_PCG_LS algorithm first computes a quadratic surrogate, $\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)})$, for the likelihood part at the current iterate, $\boldsymbol{\mu}^{(n,0)}$. Now, the cost function is composed of a quadratic function and the penalty function, which is typically non-quadratic. Subiterates, $\boldsymbol{\mu}^{(n,m)}$, are computed by setting up quadratic surrogates, $\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)})$, over the penalty function part. Thus, within the inner iteration, the cost function being minimized is completely quadratic. Now, the descent direction is computed using the PCG technique defined in (2.12). Even though the previous descent direction is from a different cost function, no significant degradation in performance is observed. Now, a line search (2.13) is used to obtain the new iterate, $\boldsymbol{\mu}^{(n,m+1)}$. A line search on a quadratic function along a descent direction can be done analytically in one step and is shown below. The algorithm is summarized as follows :

```
1.          $\boldsymbol{\mu}^{(0,0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.            for n=0,1,2,...,$N_1$-1
2.1                Compute $\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)})$
2.2                for m=0,1,2,...,$N_2$-1
2.2.1                  Compute $\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)})$
2.2.2                  $\phi^{(n,m)}(\boldsymbol{\mu}) \triangleq \phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)})$
2.2.3                  Compute descent direction $\boldsymbol{d}^{(n,m)}$ using PCG
2.2.4                  Compute $\alpha^*$ (see below).
2.2.5                  $\boldsymbol{\mu}^{(n,m+1)} = \boldsymbol{\mu}^{(n,m)} + \alpha^* \boldsymbol{d}^{(n,m)}$
2.2'               end
2.3                $\boldsymbol{\mu}^{(n+1,0)} = \boldsymbol{\mu}^{(n,N_2)}$
2'          end
```

We call the two-step surrogate, $\phi^{(n,m)}(\boldsymbol{\mu})$, the nested surrogate. We show in Section A.1 that a reduction in the value of the nested surrogate causes a reduction in the value of the objective function $\Phi(\boldsymbol{\mu})$. The total number of forward and back projection operations is $2 \times N_1 \times N_2$.

Computing α^*

The quadratic surrogates $\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)})$ and $\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)})$ can be written as :

$$\begin{aligned}
\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)}) &= \sum_{i=1}^{n_d} q_{Li}([\boldsymbol{G}\boldsymbol{\mu}]_i; [\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}]_i), \\
q_{Li}(t; s) &= h_i(s) + \dot{h}_i(s)(t-s) + \tfrac{1}{2}\breve{c}_i(s)(t-s)^2, \\
\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)}) &= \sum_{k=1}^{K} q_{Rk}([\boldsymbol{C}\boldsymbol{\mu}]_k; [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k), \\
q_{Rk}(t; s) &= \psi_k(s) + \dot{\psi}_k(s)(t-s) + \tfrac{1}{2}\breve{c}_k(s)(t-s)^2.
\end{aligned}$$

Thus, the line search in the nested surrogate can be written as :

$$\alpha^* = \text{arg min}_{\alpha \geq 0} \phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}),$$

$$\phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}) = \phi_L(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,m)}).$$

Substituting the definitions of $\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)})$ and $\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)})$ yields,

$$\begin{aligned}
\phi_L(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,0)}) = \sum_{i=1}^{n_d} \Bigg[& h_i([\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}]_i) \\
& + \dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}]_i)\Big([\boldsymbol{G}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(n,0)})]_i + \alpha [\boldsymbol{G}\boldsymbol{d}^{(n,m)}]_i\Big) \\
& + \tfrac{1}{2}\breve{c}_i\Big([\boldsymbol{G}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(n,0)})]_i + \alpha [\boldsymbol{G}\boldsymbol{d}^{(n,m)}]_i\Big)^2 \Bigg],
\end{aligned}$$

$$\begin{aligned}
\text{and } \phi_R(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,m)}) = \sum_{k=1}^{K} \Bigg[& \psi_k([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k) + \dot{\psi}_k([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)[\boldsymbol{C}\boldsymbol{d}^{(n,m)}]_k \alpha \\
& + \tfrac{1}{2}\breve{c}_k([\boldsymbol{C}\boldsymbol{d}^{(n,m)}]_k)^2 \alpha^2 \Bigg].
\end{aligned}$$

Substituting $\phi_L(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,0)})$ and $\phi_R(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}; \boldsymbol{\mu}^{(n,m)})$ in the expression for the $\phi^{(n,m)}(\boldsymbol{\mu})$ in line search above and differentiating with respect to α, we get,

$$\begin{aligned}\frac{d}{d\alpha}\phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}) = \sum_{i=1}^{n_d} \Bigg[& \Big\{ \dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}]_i) + \breve{c}_i [\boldsymbol{G}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(n,0)})]_i \Big\} [\boldsymbol{G}\boldsymbol{d}^{(n,m)}]_i \\ & + \breve{c}_i [\boldsymbol{G}\boldsymbol{d}^{(n,m)}]_i^2 \alpha \Bigg] \\ + \sum_{k=1}^{K} \Bigg[& \dot{\psi}_k([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)[\boldsymbol{C}\boldsymbol{d}^{(n,m)}]_k + \breve{c}_k([\boldsymbol{C}\boldsymbol{d}^{(n,m)}]_k)^2 \alpha \Bigg].\end{aligned}$$

Using a more compact notation, we write,

$$\begin{aligned}\frac{d}{d\alpha}\phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}) &= \Big[\dot{\boldsymbol{h}}(\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}) + \mathcal{D}(\breve{c}_i)\big(\boldsymbol{G}\boldsymbol{\mu}^{(n,m)} - \boldsymbol{G}\boldsymbol{\mu}^{(n,0)}\big) \Big]^T (\boldsymbol{G}\boldsymbol{d}^{(n,m)}) \\ &+ (\boldsymbol{G}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_i)(\boldsymbol{G}\boldsymbol{d}^{(n,m)})\alpha \\ &+ \dot{\boldsymbol{\psi}}^T(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})(\boldsymbol{C}\boldsymbol{d}^{(n,m)}) + (\boldsymbol{C}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_k)(\boldsymbol{C}\boldsymbol{d}^{(n,m)})\alpha.\end{aligned}$$

Equating the above to 0 yields the following expression for the step size:

(6.2)

$$\alpha^* = -\frac{\big\{ \dot{\boldsymbol{h}}^T(\boldsymbol{G}\boldsymbol{\mu}^{(n,0)}) + (\boldsymbol{G}\boldsymbol{\mu}^{(n,m)} - \boldsymbol{G}\boldsymbol{\mu}^{(n,0)})^T \mathcal{D}(\breve{c}_i) \big\} (\boldsymbol{G}\boldsymbol{d}^{(n,m)}) + \dot{\boldsymbol{\psi}}^T(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})(\boldsymbol{C}\boldsymbol{d}^{(n,m)})}{(\boldsymbol{G}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_i)(\boldsymbol{G}\boldsymbol{d}^{(n,m)}) + (\boldsymbol{C}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_k)(\boldsymbol{C}\boldsymbol{d}^{(n,m)})}.$$

Also, note that

$$\begin{aligned}\frac{d^2}{d^2\alpha^2}\phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)} + \alpha \boldsymbol{d}^{(n,m)}) &= (\boldsymbol{G}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_i)(\boldsymbol{G}\boldsymbol{d}^{(n,m)}) + (\boldsymbol{C}\boldsymbol{d}^{(n,m)})^T \mathcal{D}(\breve{c}_k)(\boldsymbol{C}\boldsymbol{d}^{(n,m)}), \\ &\geq 0.\end{aligned}$$

This is because the curvatures $\breve{c}_i$ and $\breve{c}_k$ are non-negative.

This algorithm is monotonic because quadratic surrogate functions are setup every iteration and a reduction in the quadratic surrogate function value results in the reduction of the value of the original cost function. But, the main advantage of this algorithm is that it could be used to achieve a minimum of non-quadratic cost functions. Because, at every iteration and subiteration, the quadratic surrogates majorize the cost function over the entire parameter space.

6.1.4 Preconditioners

Convergence rates of PCG algorithms can be increased by using preconditioners. A preconditioner is a matrix or an operator that approximates the inverse of the Hessian of the original cost function or its surrogate. Preconditioners have been known in numerical methods literature and used for the solution of a system of equations. For example, limited-memory BFGS matrices are used to approximate the Hessian [10]. However, preconditioners can have non-trivial memory requirements and a trade-off has to be considered between convergence rate improvement and memory requirements. From a memory view point, those preconditioners that require a storage of about the size the reconstruction volume are most appealing to be studied initially. Diagonal and Fourier pre-conditioners meet the above requirements. These have been derived in [25].

Preconditioners for cost functions with space-variant penalties described in [26] have been implemented for fan-beam systems and show considerable promise (see Section 6.3). These preconditioners can be improved by using properties of the fan-beam system model studied in [57]. Preconditioners for use when space-invariant penalties are used are described in [25].

Various preconditioners have been described in literature. A non-Fourier low-memory preconditioner was given by Vogel [66]. Preconditioners for Toeplitz systems have been considered by Chan [11]. Various preconditioners using the Sherman-Morrison formula have also been suggested [51, 55]. Other possible preconditioners with possibly higher memory requirements that could be considered are banded, sparse and slant-stack based. Various image restoration techniques could be employed as preconditioners. This fact is implied by the fact that Fourier and diagonal pre-conditioners do not require the explicit use of the system model.

6.1.5 Comparison between PCG_LS_QS and QS_PCG_LS

We compared the PCG_LS_QS and QS_PCG_LS algorithms for a fan-beam simulated dataset with Poisson negative log-likelihood (see (2.5)) and a space-variant penalty function [26]. The preconditioner of [25, Eq.(32)] was used. The variation of the cost function value with iteration number is shown in Fig. 6.1. Observe that the PCG_LS_QS has a faster convergence rate than QS_PCG_LS in the initial iterations. Since we focused on the convergence rate in initial iterations, we avoided the optimal curvatures as their computation involves an extra back-projection. We used precomputed curvatures instead [1].

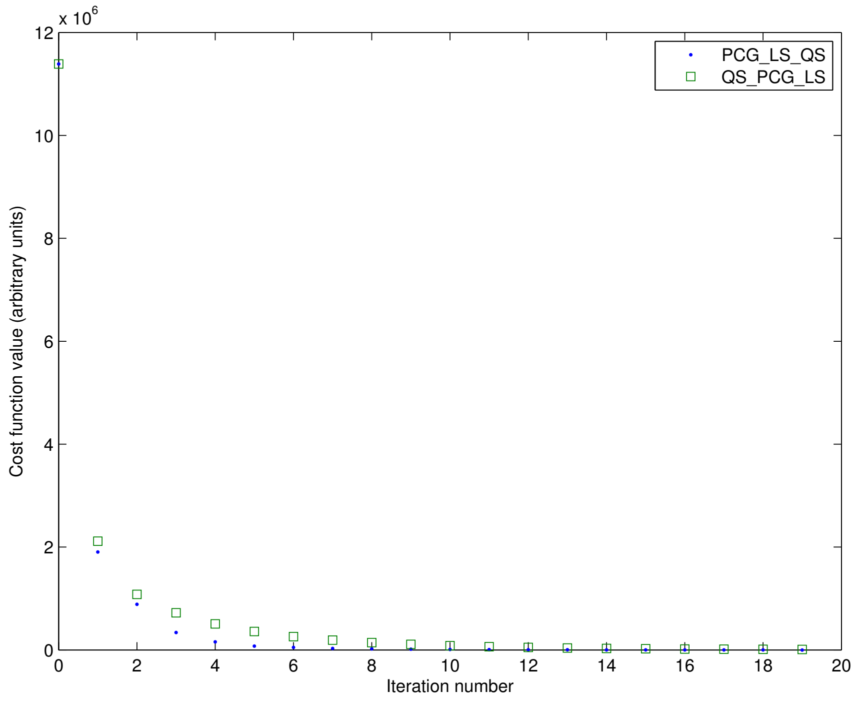


Figure 6.1: Convergence rate comparison between PCG_LS_QS and QS_PCG_LS algorithms

6.2 Ordered-subset algorithms

6.2.1 OS_SPS algorithm for an unregularized cost function

The minimization of a regularized negative log-likelihood cost function using OS and separable paraboloidal surrogates (SPS) was described in [2]. We first focus on the surrogate for the negative like-likelihood part. This will motivate the PWLS_OS_SPS algorithm in Section 6.2.2. This section also helps motivate the derivation of the OS_PCG_LS algorithm in Section 8.1. The separable surrogate for the likelihood part is derived as follows. First, a quadratic surrogate is setup at the current iterate using (2.11). The curvatures for the Poisson likelihood are computed using precomputed or optimal curvatures [1]. The negative likelihood in the WLS case is already in the quadratic form. Next, the gradient computation is approximated using OS. Due to the OS approximation to the gradient, the gradient computation time is reduced by a factor equal to the number of subsets, N_s. Finally, De Pierro's trick [15] allows us to construct a separable quadratic cost function. Without the De Pierro's trick it would not be possible to utilize the savings afforded by OS approximation of the gradient. One would expect that the convergence would be slowed by separating a quadratic function into a sum of quadratic functions, each of which depends on a single voxel. In practice, for a medium to large number of subsets, OS algorithms for X-ray CT have faster convergence rates than PCG. This fact is also demonstrated in Section 6.3.

The above procedure is summarized as follows :

$$\begin{aligned}
\phi_L(\boldsymbol{\mu};\boldsymbol{\mu}^{(n)}) &= \sum_{i=1}^{n_d} q_{L,i}(t_i;s_i) \\
&= \sum_{i=1}^{n_d} \dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(n)}]_i)[\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})]_i + \sum_{i=1}^{n_d} \tfrac{1}{2}\breve{c}_i^{(n)}[\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})]_i^2, \text{(from (2.11))} \\
&= (\boldsymbol{G}^T\dot{\boldsymbol{h}}^{(n)})^T(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})^T\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}), \\
&\approx (N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)})^T(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})^T\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}), \\
&\qquad\qquad \text{(by OS approximation to the gradient)} \\
&\leq (N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)})^T(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})^T\mathcal{D}(\sum_{i=1}^{n_d}\breve{c}_i^{(n)}\boldsymbol{G}_i|\boldsymbol{G}_{ij}|)(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}), \\
&\qquad\qquad \text{(by DePierro's trick)}
\end{aligned}$$

where, $\boldsymbol{G}_i = \sum_{j=1}^{n_p} |\boldsymbol{G}_{ij}|$, N_s is the number of subsets and $\boldsymbol{G}_r$ is the system model for the r^{th} subset. The separable cost function derived above permits the expression for a pixel update :

$$\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} - \mathcal{D}(\frac{1}{\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|}) N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)}. \tag{6.3}$$

Note that in the WLS case, $\breve{c}_i^{(n)} = w_i$, and the denominator, $\mathcal{D}(\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|) = \mathcal{D}(\sum_{i=1}^{n_d} w_i \boldsymbol{G}_i |\boldsymbol{G}_{ij}|)$. Thus, the denominator needs to be computed only once. If N_1 is the total number of iterations then the total number of equivalent forward and back projections is $2 \times N_1$.

Imposing the non-negativity constraint

The above surrogate can be used to monotonically reduce (for $N_s = 1$) the value of the cost function. This statement holds true when the non-negativity constraint is imposed, *i.e.*, when a minimizer of the cost function in the non-negative orthant is sought. The above surrogate is separable in the elements of the parameter vector, and can be expressed generally as follows: $\sum_{j=1}^{n_p} \frac{1}{2} a_j (\mu_j - b_j)^2$, where, a_j, b_j are constants. The unconstrained minimizer would be just $[b_j]_{j=1,\ldots,n_p}$. Imposing the non-negativity constraint on each element, we obtain the minimizer as $\mu_j^* = \underset{\mu_j \geq 0}{\arg\min} \frac{1}{2} a_j (\mu_j - b_j)^2$. This problem is easy enough to be solved graphically as: $\mu_j^* = \begin{cases} b_j, & b_j \geq 0, \\ 0, & b_j < 0, \end{cases}$, *i.e.*, $\mu_j^* = [b_j]_+$. Karush-Kuhn Tucker (KKT) conditions can also be used to yield the same result. Thus, the non-negative minimizer of the separable surrogate can be derived as:

$$\boldsymbol{\mu}^{(n+1)} = [\boldsymbol{\mu}^{(n)} - \mathcal{D}(\frac{1}{\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|}) N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)}]_+. \tag{6.4}$$

6.2.2 PWLS_OS_SPS algorithm

The surrogate for the likelihood part is derived in Section 6.2.1 as :

$$\phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) = (N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T \mathcal{D}_L^{(n)} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}),$$

$$\mathcal{D}_L^{(n)} \triangleq \mathcal{D}(\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|).$$

Now the surrogate function is the sum of a quadratic function and the penalty function, which is typically non-quadratic. We use the nested surrogates idea of Section 6.1.3 here. The surrogate for the penalty part is made separable and depends on the sub-iterate $\boldsymbol{\mu}^{(n,m)}$. It is derived as follows :

(6.5)

$$\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)}) = \sum_{k=1}^{n_k} q_{R,k}(t_k; s_k) = \sum_{k=1}^{n_k} \dot{\psi}([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)[\boldsymbol{C}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})]_k + \tfrac{1}{2} \breve{c}_k^{(n,m)} ([\boldsymbol{C}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})]_k)^2,$$

(6.6)

$$= (\boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T \boldsymbol{C}^T \mathcal{D}(\breve{c}_k^{(n,m)}) \boldsymbol{C} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}),$$

$$\leq (\boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T \mathcal{D}(\sum_{k=1}^{n_k} \breve{c}_k^{(n,m)} \boldsymbol{C}_k |\boldsymbol{C}_{kj}|)(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}),$$

(by De Pierro's trick)

$$= (\boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T \mathcal{D}_R^{(n,m)} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}).$$

The overall surrogate function (ignoring the constant terms) to be optimized is thus written as :

$$\phi^{(n,m)}(\boldsymbol{\mu}) = \phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) + \beta \phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)}),$$

$$= (N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)} + \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)})^T \boldsymbol{\mu} + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T \mathcal{D}_L^{(n)} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T (\beta \mathcal{D}_R^{(n,m)}) (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}).$$

Differentiating the above with respect to $\boldsymbol{\mu}$ and equating the gradient the zero, we get the update on the image:

$$
\begin{aligned}
&(N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)} + \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}) + \mathcal{D}_L^{(n)}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + (\beta \mathcal{D}_R^{(n,m)})(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) = 0 \\
&\Rightarrow \boldsymbol{\mu}^{(n,m+1)} = \frac{\mathcal{D}_L^{(n)}}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} \boldsymbol{\mu}^{(n)} + \frac{\beta \mathcal{D}_R^{(n,m)}}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} \boldsymbol{\mu}^{(n,m)} \\
&\qquad\qquad - \frac{1}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} (N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)} + \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}), \\
&\qquad = \frac{1}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} ((\beta \mathcal{D}_R^{(n,m)} \boldsymbol{\mu}^{(n,m)} - \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}) + (\mathcal{D}_L^{(n)} \boldsymbol{\mu}^{(n)} - N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})), \\
&\qquad = \frac{1}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} ((\beta \mathcal{D}_R^{(n,m)} \boldsymbol{\mu}^{(n,m)} - \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}) + \boldsymbol{z}^{(n)}),
\end{aligned}
$$

$$
\boldsymbol{z}^{(n)} \triangleq \mathcal{D}_L^{(n)} \boldsymbol{\mu}^{(n)} - N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)}. \tag{6.7}
$$

Using similar arguments for the non-negativity constraint as in Section 6.2.1, we obtain the non-negatively constrained update as:

$$
\boldsymbol{\mu}^{(n,m+1)} = [\frac{1}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}} ((\beta \mathcal{D}_R^{(n,m)} \boldsymbol{\mu}^{(n,m)} - \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}) + \boldsymbol{z}^{(n)})]_+. \tag{6.8}
$$

The pseudo code for the serial PWLS_OS_SPS is :

```
1.          $\boldsymbol{\mu}^{(0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.          for n=0,1,2,...,$N_1 \times N_s$-1
2.1             Choose subset $r$ = bit_reverse($n \bmod N_s$)
2.2             Compute $\boldsymbol{z}^{(n)} = \mathcal{D}_L^{(n)} \boldsymbol{\mu}^{(n)} - N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)}$.
2.3             for m=0,1,2,...,$N_2$-1
2.3.1             Compute
                  $$\boldsymbol{\mu}^{(n,m+1)} = [\frac{1}{\mathcal{D}_L^{(n)} + \beta \mathcal{D}_R^{(n,m)}}((\beta \mathcal{D}_R^{(n,m)} \boldsymbol{\mu}^{(n,m)} - \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}^{(n,m)}) + \boldsymbol{z}^{(n)})]_+$$
2.3'            end
2.4             $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n,N_2)}$
2'          end
```

The total number of equivalent forward and back projection operations is $2 \times N_1$.

6.3 Comparison between PWLS_OS_SPS and PWLS_PLQ

In the following simulations, the convergence rate of OS and PCG based algorithms are compared. Fan-beam geometry similar to the center row of detectors of GE Lightspeed VCT scanner was used. The sinogram has 888 bins in each of the 984 angles and the reconstruction grid size is 512×512. The blank-scan counts are 2×10^5 per ray, $r_i = 0$ and the WLS cost function was used. The regularization was space-variant, non-quadratic (hyperbola with $\delta = 50$HU - see Table 2.1) and the value of β is chosen so as to target a resolution at the center of 1.7 pixels. The preconditioners considered here are diagonal [25, Eq.(8)], circ1 [25, M_0] and circ1_cdc [25, Eq.(32)]. The average of reconstructions (1000 iterations) from 4 PCG algorithms (three use the above preconditioners and a PCG reconstruction without using a preconditioner) is used as the final

	$L_1^{avg} < 1$HU	$L_\infty < 1$HU
CG-none	70	328
CG-circ1	50	243
CG-diag	34	165
CG-circ1-cdc	16	75
SPS-OS-41	33	-
SPS-OS-246	-	-

Table 6.1: Number of iterations required to satisfy convergence criteria

image to compare convergence rates. Two OS algorithms, one using 41 subsets and the other using 246 subsets, are compared along side the PCG algorithms. Two metrics are used to compute the distance between the iterates and the final image obtained above: $L_1^{avg}(\boldsymbol{\mu}, \boldsymbol{\mu}^*) = \sum_{j=1}^{n_p} |\mu_j - \mu_j^*|/n_p$ and $L_\infty(\boldsymbol{\mu}, \boldsymbol{\mu}^*) = \max\{|\mu_j - \mu_j^*| : j = 1, \ldots, n_p\}$.

Fig. 6.2 shows that both PCG and OS based methods produce acceptable reconstructions at the window level of 400HU. From Fig. 6.3 and Fig. 6.4, we observe that the OS algorithms reduce the cost function value faster than PCG algorithms in the initial iterations. However, from Table 6.1 we find that the PCG algorithm with circ1_cdc preconditioner meets both $L_1^{avg} < 1$HU and $L_\infty < 1$HU convergence criteria in the fewest number of iterations. The OS algorithms become non-monotonic and are unable to get close to the final image. This fact is also evident from Fig. 6.3. Also note from Fig. 6.4 that there is a tradeoff between the number of subsets used and convergence rate in OS algorithms. Fig. 6.4 and Fig. 6.3 also suggest a tradeoff between number of subsets and the iteration at which an OS algorithm becomes non-monotonic. These plots motivate a hybrid strategy in these simulations. First, we would run the OS algorithm with 246 subsets for 30 iterations and create an intermediate image. And then we would use to the PCG algorithm, initialized with this intermediate image, that uses the circ1_cdc preconditioner till image quality is satisfactory or convergence criteria are met.

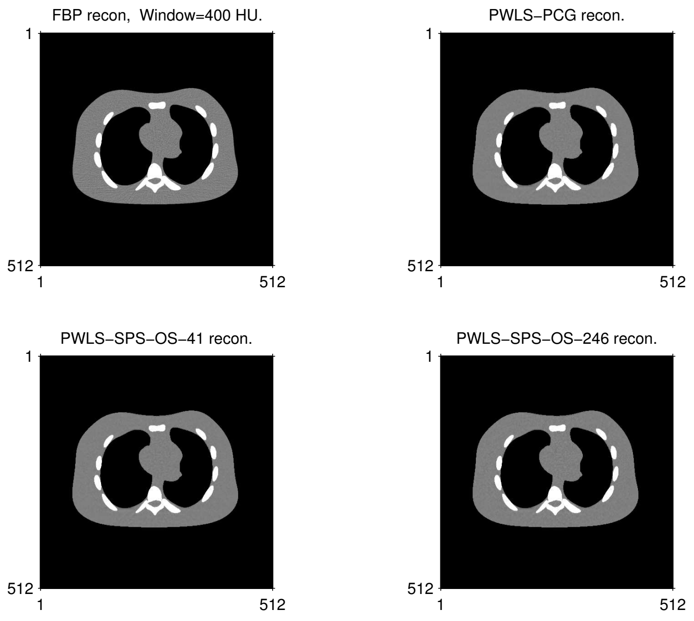


Figure 6.2: Reconstructions from statistical algorithms. Window = 400HU.

6.4 Summary

Monotonic algorithms are important because they are the first step towards finding convergent algorithms. Non-monotonic algorithms stop reducing the value of the cost function after a few iterations, and thus may not be useful for some applications. Between OS and PCG, only PCG based algorithms can achieve monotonicity. PCG algorithms known till now did not guarantee monotonicity for non-quadratic cost functions, and this void was filled in this chapter with the derivation of QS_PCG_LS. It was also showed that monotonicity can be achieved without sacrificing convergence rate. While monotonic algorithms are required for achieving the best possible reconstruction a cost function can offer, high convergence rate algorithms are required to make statistical methods practical. Various OS based methods and PCG methods with different preconditioners were compared, and it was found that a higher convergence rate can be achieved using OS based methods when the initial image is FBP.

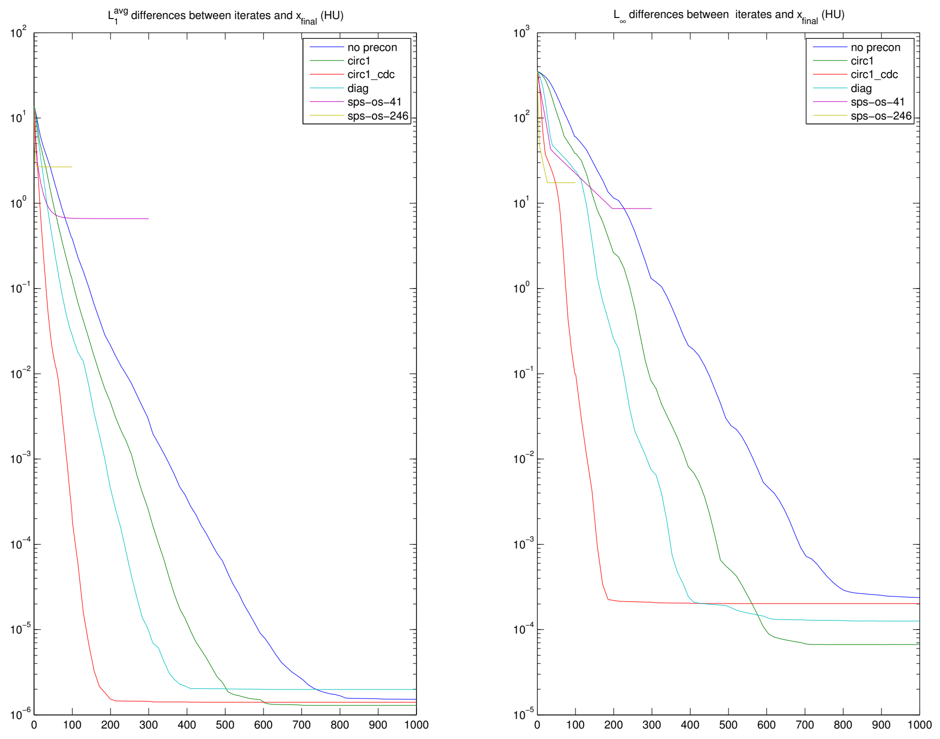


Figure 6.3: Difference plots between statistical reconstructions and final reconstruction, 1000 iterations. *Left*: L_1^{avg}, *Right*: L_∞.

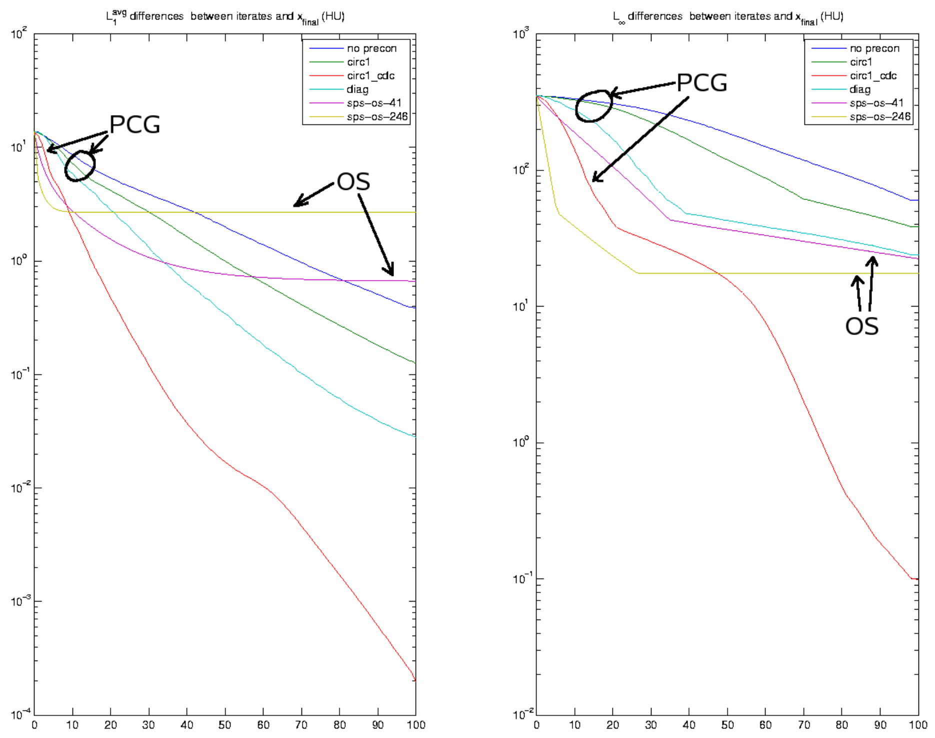


Figure 6.4: Difference plots between statistical reconstructions and final reconstruction, first 100 iterations. *Left*: L_1^{avg}, *Right*: L_∞.

CHAPTER 7

Algorithm acceleration using hybrid-algorithm and parallel-computation approaches

Statistical image reconstruction methods need to be accelerated to produce reconstructed images from a X-ray CT scanner in a reasonable amount of time. Two approaches towards algorithm acceleration are discussed in this chapter: hybrid-algorithm (Section 7.1) and parallel-computation (Section 7.2). The hybrid-algorithm approach aims to create an iterative algorithm with a higher convergence-rate by combining existing algorithms. The goal of the parallel-computation approach is to reduce execution times of existing algorithms by distributing their mathematical computations among the individual processors of a parallel-computer. The ICD algorithm is being used by our colleagues at General Electric Healthcare Inc. to reconstruct images from the data of the MSCT scanner, GE Lightspeed VCT. We investigate whether the ICD's compute time can be reduced by the OS-ICD hybrid algorithm (proposed in Section 7.1) running on a parallel-computer. In Section 7.2.1, we first discuss the derivation of the Parallel PWLS_OS_SPS algorithm from the PWLS_OS_SPS algorithm of Section 6.2.2. In Section 7.2.2, we then describe the outline of the implementation of the Parallel PWLS_OS_SPS algorithm. In Section 7.3, we present the results from reconstructing a data set obtained from the GE Lightspeed VCT scanner using the OS-ICD hybrid algorithm.

7.1 Hybrid-algorithm approach

7.1.1 Basic principle

Empirical observation has shown that different algorithms have different convergence rates depending on how far the current iterate is from the minimum/minima of the cost function. This difference in convergence rates can be exploited to combine two algorithms into a hybrid algorithm as follows. Initialize the first algorithm with the initial image (assume that the initial image is far away from the minimum/minima of the cost function) and run it for a few iterations to produce an intermediate reconstructed image. Then, initialize the second algorithm with the intermediate image and run it until convergence criteria are met. This combination method would reduce the overall compute time if two conditions are satisfied. Firstly, the first algorithm should be faster than the second when initialized by an image far away from the minimum/minima. And secondly, the second algorithm should be faster than the first when initialized by an image closer to the minimum/minima of the cost function. A different combination method would be to take the "numerical tricks" of each algorithm and somehow combine those tricks to produce a hybrid-algorithm. The second method is used to design a hybrid OS-PCG algorithm in Section 8.1, and will not be explored in this chapter.

7.1.2 OS-ICD hybrid algorithm

Three algorithms have been used in this study for the image reconstruction in X-ray CT: OS, PCG, and ICD. The ICD algorithm has been used by our collaborators extensively to produce image reconstructions, and with a high convergence rate in many cases [61]. However, in some cases, intermediate ICD iterates overshoot the the minimum/minima of the cost function, resulting in a large number of iterations to meet the convergence criteria. These overshoots typically happen when the current iterate is far away from the minimum/minima; for example, when regions of the image are at, say, 0HU, while the reconstructed values in those regions would typically be 1000HU. The two candidate algorithms to counter the overshoot problem are OS and PCG. Chapter 6 showed that, starting from FBP, OS reduces the cost function value faster than PCG. Even though those comparisons are for fan-beam scans, empirical convergence rates of PCG are not expected to be higher than OS in axial and helical cone-beam scans. The reason is the close geometric similarity between fan-beam and axial and helical cone-beam scans. Thus, the OS-ICD hybrid algorithm is a better candidate than PCG-ICD hybrid algorithm for beating the compute time of a pure ICD algorithm.

7.2 Parallel-computation approach

The overall computation time of an iterative statistical image reconstruction algorithm is the product of number of iterations and the per-iteration compute time. The number of iterations can be reduced by using algorithms with faster convergence rates; this is accomplished by the hybrid-algorithm approach in Section 7.1. Reduction of per-iteration compute time can be accomplished by using faster computer hardware and software. Only algorithm acceleration using faster computer hardware is explored here. Use of faster computer software, *e.g.*, assembly language programming, to accelerate algorithms is beyond the scope of this thesis. One way to build a faster computer is to aggregate multiple microprocessors into a parallel-computer. The mathematical computations of the algorithm can then be distributed among the processors to run concurrently. An other way to create a faster computer is to use specialized hardware like graphics processor, cell processor, and field programmable gate-array (FPGA). In this thesis, we focus on the use of parallel-computers to speed up iterative algorithms. This thesis deals with the distribution of the computations of the PWLS_OS_SPS algorithm (Section 6.2.2) among the individual processors of a parallel-computer. The resulting algorithm is called the Parallel PWLS_OS_SPS algorithm (Section 7.2.1).

Recent work on algorithm acceleration using parallel computers can be found in [42] and [41]. In [42], the authors reported a speedup of 30 in reconstructing an image of size $512 \times 512 \times 400$ for a C-arm CT scanner. The authors used two strategies for parallelism :

1. Share the reconstruction volume among processors and distribute the sinogram rays ([42, OSC-ang]), and
2. Distribute the reconstruction volume among processors and share the sinogram rays ([42, OSC-vol]).

In [41], the authors describe the implementation of the AM-OS algorithm for MSCT scanner data on a parallel-computer with 64 processors.

7.2.1 Parallel PWLS_OS_SPS

The PWLS_OS_SPS algorithm of Section 6.2.2 is repeated here for convenience.

```
1.        $\boldsymbol{\mu}^{(0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.        for n=0,1,2,...,$N_1 \times N_s$-1
2.1           Choose subset $r$ = bit_reverse($n \bmod N_1$)
2.2           Compute $\boldsymbol{z}^{(n)} = \mathcal{D}_L \boldsymbol{\mu}^{(n)} - N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r(\boldsymbol{G}_r \boldsymbol{\mu}^{(n)})$.
2.3           for m=0,1,2,...,$N_2$-1
2.3.1             Compute
                  $\boldsymbol{\mu}^{(n,m+1)} = [\frac{(\beta \mathcal{D}_R(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})\boldsymbol{\mu}^{(n,m)} - \beta \boldsymbol{C}^T \dot{\boldsymbol{\psi}}(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})) + \boldsymbol{z}^{(n)}}{\mathcal{D}_L + \beta \mathcal{D}_R(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})}]_+.$
2.3'          end
2.4           $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n,N_2)}$
2'        end
```

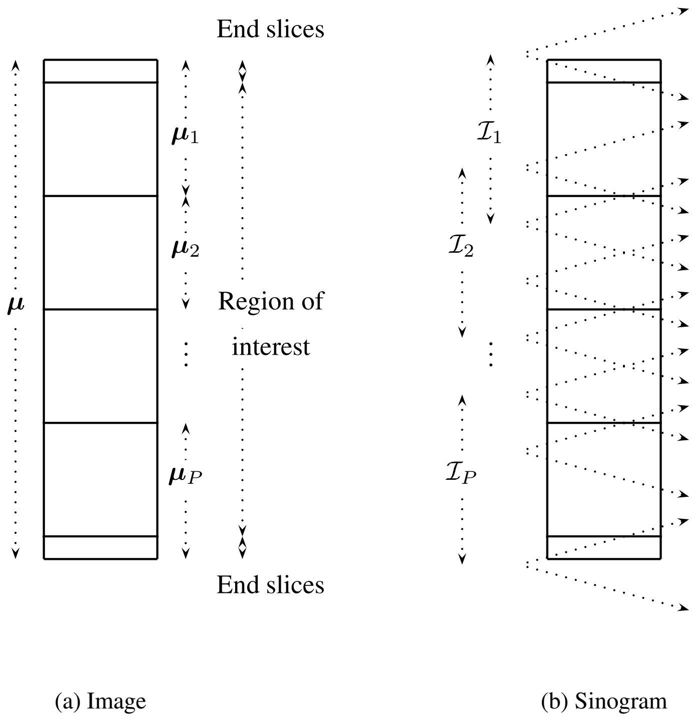


Figure 7.1: Division of image and sinogram spaces among $P = 4$ processors. Longitudinal *i.e.*, z-axis, view is presented here.

Two changes are made from Section 6.2.2. First, $\mathcal{D}_L^{(n)}$ is independent of $\boldsymbol{\mu}^{(n)}$ for the WLS negative log-likelihood and can be replaced by $\mathcal{D}_L$. And second, $\mathcal{D}_R^{(n,m)}$ is replaced by $\mathcal{D}_R(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})$ to show explicitly the variables it depends on.

The method used here to distribute the computations of PWLS_OS_SPS among P processors is as follows:

1. Divide the image voxel variables of $\boldsymbol{\mu}$ into P disjoint subsets. Further assume that each subset is a stack of contiguous slices. This division is denoted here as $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_P \end{bmatrix}$. Also enforce the following constraints. First, only the pth processor may update variables of $\boldsymbol{\mu}_p$. And second, any processor may read from any component of

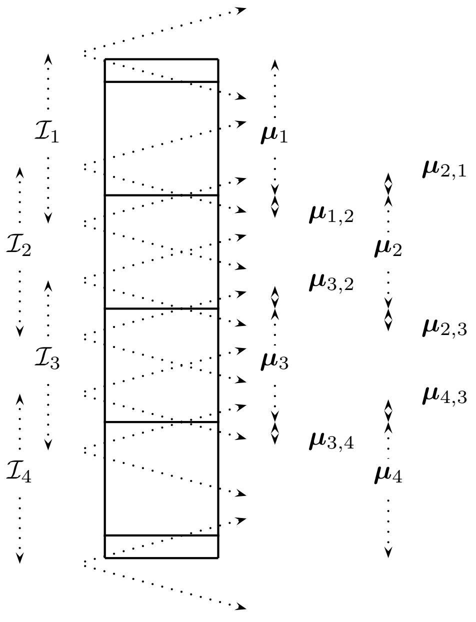


Figure 7.2: Illustration of $\boldsymbol{\mu}_{p,p'}$ for $P = 4$ processors.

$\boldsymbol{\mu}$. This, in effect, means that, pth processor is "responsible" for $\boldsymbol{\mu}_p$. See Fig. 7.1(a). In Fig. 7.1(a), the region of interest *i.e.*, the set of slices to be scanned, is specified by the radiologist at the beginning of the scan. All rays passing through the region of interest are collected (see Fig. 7.1(b)). Therefore, the region of interest is as well-sampled as possible. The slices needed to compute forward and back projections for rays that do not completely lie in the region of interest are called extra slices. Thus, extra slices have to be estimated also for the statistical method to work, even though enough data for them is not available. This is a consequence of the long-object problem for statistical methods.

2. Divide the observed data *i.e.*, sinogram, into P subsets, $\mathcal{I}_p, p = 1, \ldots, P$, as follows. Each ray of $\mathcal{I}_p$ must pass through atleast one voxel of $\boldsymbol{\mu}_p$. $\mathcal{I}_p$ can not be disjoint here because of the oblique nature of many rays in the cone-beam geometry. Also, if the subsets $\mathcal{I}_p, p = 1, \ldots, P$ were disjoint, we would have P independent problems. See

Fig. 7.1(b).

The division of observations described here helps us to understand the problem better. However, the direct implementation of the parallel PWLS_OS_SPS algorithm based on this division causes two or more processors to repeat some numerical computations. The implementation of the parallel PWLS_OS_SPS algorithm must avoid such repetition. Our implementation avoids this repetition (Section 7.2.2).

3. Now, during the computation process, each processor would forward project $\boldsymbol{\mu}_p$ into rays (or, sinogram bins) of $\mathcal{I}_p$. Since one or more rays of $\mathcal{I}_p$ would pass through neighboring image segments, image slices from neighboring image segments would be required. Denote by $\boldsymbol{\mu}_{p,p'}$, those slices of image segment p' needed to compute the forward projection of $\boldsymbol{\mu}_p$ into $\mathcal{I}_p$. Also note that, $\boldsymbol{\mu}_{p,p}$ is simply $\boldsymbol{\mu}_p$. See examples of $\boldsymbol{\mu}_{p,p'}$ for $P = 4$ in Fig. 7.2.

4. The additively-separable forms of both negative likelihood and the penalty function bring to light the following analogy: "What rays are to sinograms, voxel differences are to the superset of regularization neighborhoods". Thus, in same vein as $\boldsymbol{\mu}_{p,p'}$, define $\boldsymbol{\mu}_{p,p',pen}$.

5. The computation of $\boldsymbol{z}^{(n)}$ in step 2.2 of the above pseudo-code can be split into P components as

$$\begin{bmatrix} \boldsymbol{z}_1^{(n)} \\ \vdots \\ \boldsymbol{z}_P^{(n)} \end{bmatrix} = \mathcal{D}_L \begin{bmatrix} \boldsymbol{\mu}_1^{(n)} \\ \vdots \\ \boldsymbol{\mu}_P^{(n)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{v}_1^{(n)} \\ \vdots \\ \boldsymbol{v}_P^{(n)} \end{bmatrix}, \boldsymbol{v}^{(n)} \triangleq N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r(\boldsymbol{G}_r \boldsymbol{\mu}^{(n)}).$$

For the pth component of $\boldsymbol{z}^{(n)}$, we can write the above computation as $\boldsymbol{z}_p^{(n)} = \mathcal{D}_{L,p}\boldsymbol{\mu}_p^{(n)} - \boldsymbol{v}_p^{(n)}$, where, the diagonal matrix $\mathcal{D}_L$ is split into p diagonal matrices $\mathcal{D}_{L,p}$ in exactly the same way as $\boldsymbol{\mu}$ is split into p image components. The tricky part is the compuatation of $\boldsymbol{v}_p^{(n)}$. Note that the operator $\boldsymbol{G}_r^T$ in the equation $\boldsymbol{v}^{(n)} \triangleq N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r(\boldsymbol{G}_r \boldsymbol{\mu}^{(n)})$ maps rth subset view errors into $\boldsymbol{v}^{(n)}$. But, to compute $\boldsymbol{v}_p^{(n)}$ we only need those views from the the rth subset view that belong to $\mathcal{I}_p$. So, we only need those image slices through which the rays of $\mathcal{I}_p$ pass. These image slices are given by $\begin{bmatrix} \boldsymbol{\mu}_{p,1}^{(n)} \\ \vdots \\ \boldsymbol{\mu}_{p,P}^{(n)} \end{bmatrix}$. Thus, the computation of $\boldsymbol{v}_p^{(n)}$ can be written as $\boldsymbol{v}_p^{(n)} = N_s \Big[\boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r (\boldsymbol{G}_r \begin{bmatrix} \boldsymbol{\mu}_{p,1}^{(n)} \\ \vdots \\ \boldsymbol{\mu}_{p,P}^{(n)} \end{bmatrix}) \Big]_p$, where the notation $[\boldsymbol{\mu}]_p$ means the pth component of $\boldsymbol{\mu}$ *i.e.*, $[\boldsymbol{\mu}]_p \triangleq \boldsymbol{\mu}_p$.

6. Using similar arguments from the previous step, the image update in step 2.3.1 can be written in the "parallel" form as

$$\begin{aligned}
\boldsymbol{\mu}_p^{(n,m+1)} &= [\frac{(\beta \mathcal{D}_{R,p}(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})\boldsymbol{\mu}_p^{(n,m)} - \beta \Big[\boldsymbol{C}^T \dot{\boldsymbol{\psi}}(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})\Big]_p) + \boldsymbol{z}_p^{(n)}}{\mathcal{D}_{L,p} + \beta \mathcal{D}_{R,p}(\boldsymbol{C}\boldsymbol{\mu}^{(n,m)})}]_+, \\
&= [\frac{(\beta \mathcal{D}_{R,p}(\boldsymbol{C} \begin{bmatrix} \boldsymbol{\mu}_{p,1,pen}^{(n,m)} \\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)} \end{bmatrix})\boldsymbol{\mu}_p^{(n,m)} - \beta \Big[\boldsymbol{C}^T \dot{\boldsymbol{\psi}}(\boldsymbol{C} \begin{bmatrix} \boldsymbol{\mu}_{p,1,pen}^{(n,m)} \\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)} \end{bmatrix})\Big]_p) + \boldsymbol{z}_p^{(n)}}{\mathcal{D}_{L,p} + \beta \mathcal{D}_{R,p}(\boldsymbol{C} \begin{bmatrix} \boldsymbol{\mu}_{p,1,pen}^{(n,m)} \\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)} \end{bmatrix})}]_+.
\end{aligned}$$

The pseudo-code for the parallel PWLS_OS_SPS is as follows:

```
1.        $\boldsymbol{\mu}^{(0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.        for n=0,1,2,...,$N_1 \times N_s$-1
2.1           Choose subset $r$ = bit_reverse($n \bmod N_1$)
2.2           for p=1,...,P compute in parallel
```

$$\boldsymbol{z}_p^{(n)} = \mathcal{D}_{L,p}\boldsymbol{\mu}_p^{(n)} - N_s\Big[\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r(\boldsymbol{G}_r\begin{bmatrix}\boldsymbol{\mu}_{p,1}^{(n)}\\ \vdots \\ \boldsymbol{\mu}_{p,P}^{(n)}\end{bmatrix})\Big]_p$$

.

```
2.3           for m=0,1,2,...,$N_2$-1
2.3.1           for p=1,...,P compute in parallel
```

$$\boldsymbol{\mu}_p^{(n,m+1)} = [\frac{(\beta\mathcal{D}_{R,p}(\boldsymbol{C}\begin{bmatrix}\boldsymbol{\mu}_{p,1,pen}^{(n,m)}\\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)}\end{bmatrix})\boldsymbol{\mu}_p^{(n,m)} - \beta\Big[\boldsymbol{C}^T\dot{\boldsymbol{\psi}}(\boldsymbol{C}\begin{bmatrix}\boldsymbol{\mu}_{p,1,pen}^{(n,m)}\\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)}\end{bmatrix})\Big]_p) + \boldsymbol{z}_p^{(n)}}{\mathcal{D}_{L,p} + \beta\mathcal{D}_{R,p}(\boldsymbol{C}\begin{bmatrix}\boldsymbol{\mu}_{p,1,pen}^{(n,m)}\\ \vdots \\ \boldsymbol{\mu}_{p,P,pen}^{(n,m)}\end{bmatrix})}]_+.$$

```
2.3'          end
2.4           $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n,N_2)}$
2'        end
```

7.2.2 Implementation aspects of the Parallel PWLS_OS_SPS algorithm

The parallel PWLS_OS_SPS algorithm was derived in Section 7.2. The implementation of the algorithm in software is outlined in this section.

The implementation described here has three very useful properties:

1. Computer architecture adaptive

   The organization of processors in a computer is hierarchical, *e.g.*, a multi-processor, multi-core computer consists of many processors, and a processor consists of many cores. We call the first level of the processor hierarchy, the MPU (memory processor unit) and the second level, the BPU (base processor unit). The software implementation consists of many threads running concurrently. If the data sharing between threads (called, inter-processor communication) executing on the assigned MPUs and BPUs follows the processor hierarchy, then the utilization of memory buses can be controlled precisely. This would allow maximum utilization of resources of a given computer.

2. Memory efficient

   Dividing the problem into a large number of processors causes memory requirements to increase linearly. This is because of the creation of a large number of $\boldsymbol{\mu}_{p,p'}$ image segments. For the program to work, the memory requirements must not exceed the maximum amount of random access memory (RAM) available. The implementation described here divides the problem into only as many MPUs as permitted by the available RAM. Each MPU is then assigned BPUs to maximize processor utlilization. Thus, in presence of memory constraints, the MPU-BPU division might not follow the processor hierarchy of a given computer and, the utilization of memory buses would be sub-optimal.

3. Repetetive numerical computation avoidance

   In Section 7.2, the Parallel PWLS_OS_SPS algorithm was designed using an overlapping division of the observations, $\mathcal{I}_p, p = 1, \ldots, P$. This division is intuitive and helps in understanding the problem. But, this division causes some calculations to be

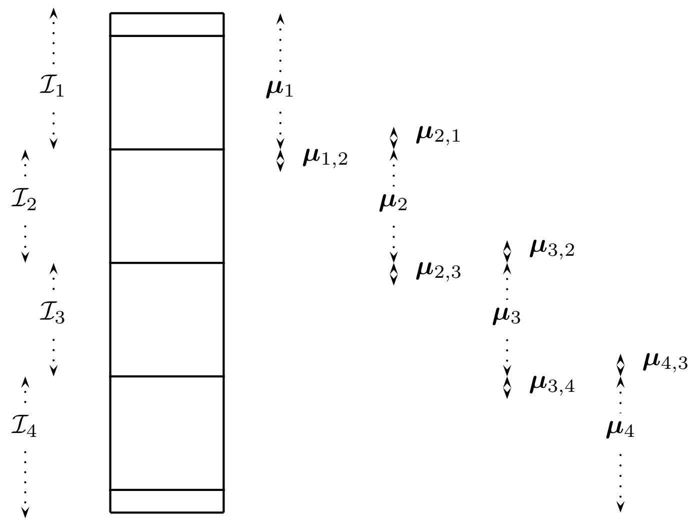


Figure 7.3: Image and observation space division for $P = 4$ MPUs

repeated by two or more processors. This implementation is designed to avoid such a repetition while still performing the computations of the Parallel PWLS_OS_SPS algorithm.

The major aspects of the software implementation are described next.

Image and observation space division

The division of the observation space among MPUs is no longer overlapping. It is disjoint (Fig. 7.3). A consequence of this disjoint division is that some of the rays passing through, say $\boldsymbol{\mu}_1$, are in $\mathcal{I}_2$. The forward and back projection operations will have to be adapted to this scheme. More on this later.

The image space is divided among MPUs just as before but the definition of $\boldsymbol{\mu}_{p,p'}$ is different. $\boldsymbol{\mu}_{p,p'}$ is that segment of $\boldsymbol{\mu}_{p'}$ which is non-zero, when a uniform sinogram of ones corresponding to $\mathcal{I}_p$ is back-projected onto $\boldsymbol{\mu}_{p'}$. The image $\boldsymbol{\mu}_p$ is represented in the code as `ownvol`$[p]$ (short for "own volume"). The image $\begin{bmatrix} \boldsymbol{\mu}_{p,1} \\ \vdots \\ \boldsymbol{\mu}_{p,P} \end{bmatrix}$ is represented in the code as `fullvol`$[p]$ (short for "full volume").

Full and subset forward projection

Since subset forward projection is a subset of the full forward projection, the image space division defined above can be retained. For simplicity, let us consider the full foward projection only; the extension to the subset forward projection is trivial.

The forward projection is divided into MPUs as follows. MPU p computes the pth component of the sinogram, $\mathcal{I}_p$. For this to happen, all image slices relevant to the computation of $\mathcal{I}_p$ will have to copied to MPU p. Thus, the processor copies the corresponding slices of $\boldsymbol{\mu}_{p'}$ into slices of $\boldsymbol{\mu}_{p,p'}$. This operation is denoted in the code as `sync` (Fig. 7.4).

The views in $\mathcal{I}_p$ are further divided into BPUs. The division of sinograms among BPUs is also disjoint. The forward projection involves reading image slices and each BPU can do the reading without coordinating with other BPUs. This is in contrast to the back-projection (described next).

Full and subset back projection

Since subset back projection is a subset of the full back projection, the image space division defined above can be retained. For simplicity, let us consider the full back projection only; the extension to the subset back projection is trivial.

The back projection is divided into MPUs as follows. MPU p computes the pth component of the image, $\boldsymbol{\mu}_p$. Now, $\boldsymbol{\mu}_p$ would require rays not only from $\mathcal{I}_p$ but also other MPUs.

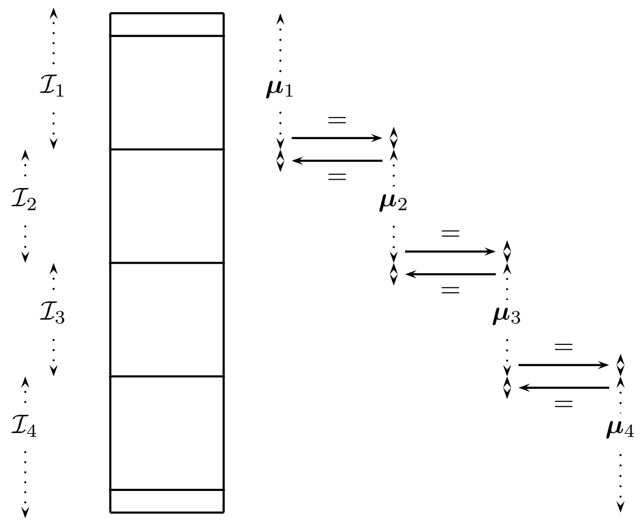


Figure 7.4: The `sync` operation.

This is accomplished as follows. Each MPU p back-projects $\mathcal{I}_p$ onto $\begin{bmatrix} \boldsymbol{\mu}_{p,1} \\ \vdots \\ \boldsymbol{\mu}_{p,P} \end{bmatrix}$. Now, the back projection of rays from $\mathcal{I}_{p'}$ onto $\boldsymbol{\mu}_p$ would be stored in $\boldsymbol{\mu}_{p',p}$ (this is due to the definition of $\boldsymbol{\mu}_{p,p'}$). Thus, the back projection of $\boldsymbol{\mu}_p$ is computed by adding slices of $\boldsymbol{\mu}_{p',p}$ to corresponding slices of $\boldsymbol{\mu}_p$. This operation is denoted in the code as `sum` (Fig. 7.5).

The views in $\mathcal{I}_p$ are further divided into BPUs. The division of sinograms among BPUs is also disjoint. The back projection operation involves writing to image slices and each BPU has to coordinate with other BPUs. The access control is performed using mutual exclusion locks of the pthread library. This is in contrast to forward-projection (described previously).

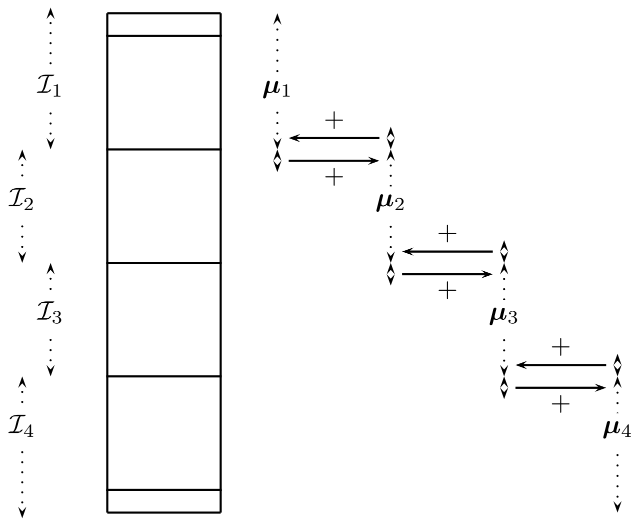


Figure 7.5: The `sum` operation.

7.3 Results

Results from image reconstructions of a helical cone-beam dataset from a MSCT scanner, GE Lightspeed, using the hybrid OS-ICD and ICD algorithms running on a parallel computer are presented here.

7.3.1 Scanner and dataset description

The GE Lightspeed scanner is a 3^{rd} generation CT scanner with a single X-ray source and an arc-detector array placed opposite each other, on a gantry.The detector-array has 64 detector-rows and 888 detector channels per row. Each detector is roughly 1mm $\times$ 1mm in size. The radius of rotation of the source and detector-array are roughly 54cm and 41cm respectively. In the current dataset, 3062 views of the object are taken with the pitch of the helix being nearly 1:1 and a source translation of nearly 4cm per rotation. The pitch of the helix is defined here as the ratio between the source translation in the longitudinal direction in one rotation of the gantry, and the width of the detector array measured at the isocenter of the gantry. 984 views are obtained in a $360°$ rotation of the gantry, giving us slightly more than 3 rotations around the patient. The raw projection-data and scanner parameters were provided by GE Healthcare.

7.3.2 Hardware and software

The parallel computer hardware used in this research is made of 4 AMD dual-core processors connected to 16GB of shared random access memory. Thus, this computer has 8 processors in the sense of the word processor used before (we count a core as a processor here). The OS algorithm was implemented in the C language and the pthreads library was used to implement parallel processing. The operating system used was 64-bit linux.

7.3.3 Reconstructed images

Two reconstructed images are produced using ICD and hybrid OS-ICD algorithms and compared with each other here. The initial image for both algorithms is an FBP image, and was provided by GE Healthcare. The image grid covers a large field-of-view in the axial plane, a circle of diameter of about 70cm. The extent of the region of interest in the longitudinal direction (*i.e.*, z-direction) is about 6cm. Due to the oblique nature of the cone-beam rays and the large cone angle of 64 slice (*i.e.*, 64 detector-row) scanners, we need to reconstruct an image grid whose extent in z-direction is about 12cm. The voxel dimensions are about 1.4mm $\times$ 1.4mm $\times$ 0.6mm, and the image grid size is $512 \times 512 \times 185$.

The cost function used here was the sum of the WLS negative log-likelihood and space-invariant non-quadratic penalty function. The parameters of the WLS negative log-likelihood were provided by GE Healthcare, which they had computed from the raw observations generated by the GE Lightspeed scanner. The potential function used initially in the penalty function was q-GGMRF (see Table 2.1). This was later replaced by the Generalized-Fair (see Table 2.1) potential function to achieve significant compute time savings without compromising image quality. Due to the 3-D nature of the image volume and the use of first-order differences in the regularization, the number of voxel neighbors employed in regularization was 26.

Fig. 7.6 shows a coronal ($x - z$) slice of the FBP image (top), and the intermediate image (bottom) produced at the end of the OS part of the hybrid OS-ICD algorithm. 12 iterations of the OS algorithm with 41 subsets and initialized by FBP were run to produce the intermediate image. Fig. 7.7 shows a coronal slice of the hybrid OS-ICD reconstruction (top) and the ICD reconstruction (bottom). The hybrid OS-ICD reconstruction was obtained when the ICD algorithm was initialized by the intermediate image obtained above. The ICD reconstruction was obtained when the ICD algorithm was initialized by FBP. The implementation of the ICD algorithm was provided by GE Healthcare, while the implementation of the OS algorithm, except the system model, was carried out here, at the University of Michigan. The system model implementation was provided by GE Healthcare.

Top slices in the FBP image are replications of the end slice. This is because top slices are not sampled fully by the helical cone-beam geometry. And, FBP algorithms do not respond well to data insufficiency. So, the last well-reconstructed slice is used for all the top slices in FBP. Further note that both statistical algorithms, ICD and hybrid OS-ICD, reconstruct more number of end-slices than FBP. Also note that the ICD algorithm does not update the image in the top and bottom slices while the OS algorithm does update them. In a given end-slice, atleast one voxel has a ray passing through it. But, the entire slice has to be included in the reconstruction with the OS algorithm to keep the implementation simple and fast. For the voxels that have no or very few rays passing through them, we inherently depend on regularization to update them. That is why end-slices are very blurred in the OS reconstruction. This blurring is observed to some extent in the ICD reconstruction also.

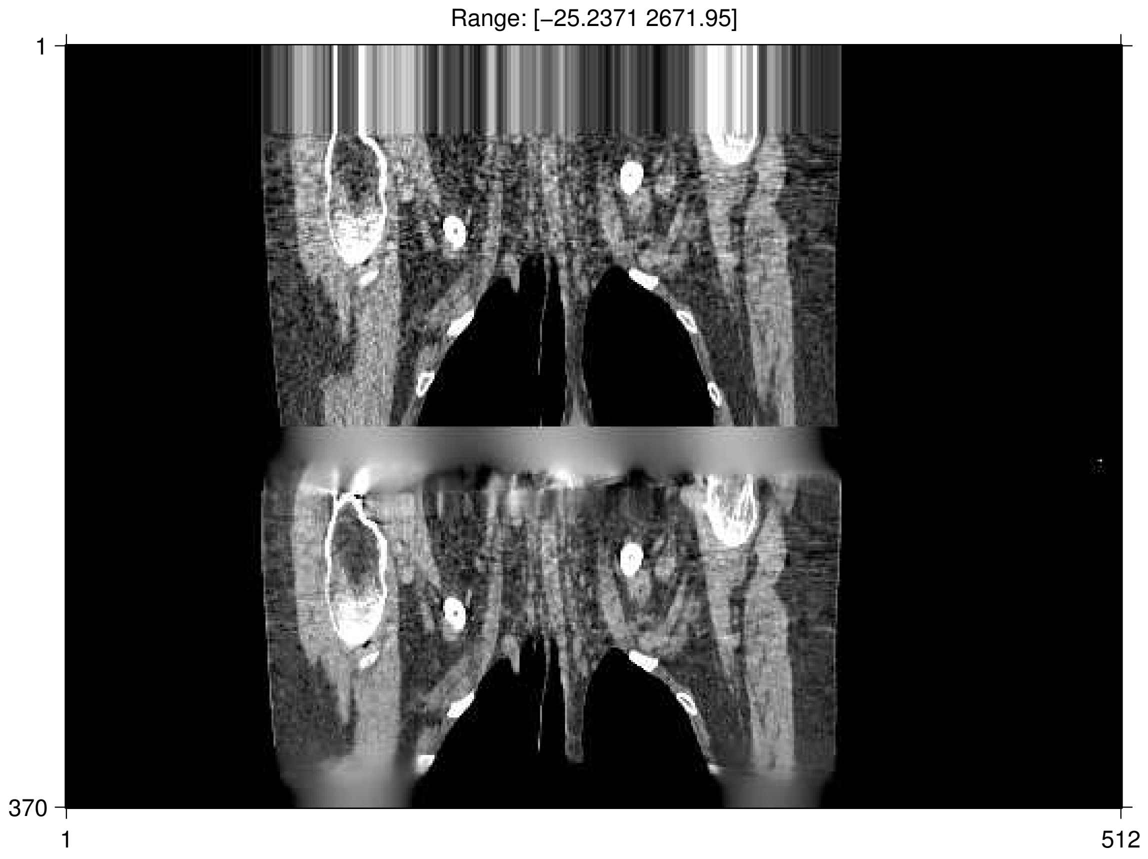


Figure 7.6: FBP image and intermediate image from hybrid OS-ICD algorithm. *Top:* Initial FBP image, and *Bottom:* Intermediate reconstruction. Image window = $[800, 1200]$ HU.

7.3.4 Compute time of the algorithms

The compute time of the algorithms depends on the hardware and the software optimization of the algorithm implementation. Software optimization is the process of changing the software implementation to make a program run faster and/or its memory footprint smaller by various methods like changing the organization of the memory allocated by the software, using assembly programming, load-balancing – adjusting the amount of number-crunching given to a particular thread *etc.*. Software optimization was done on

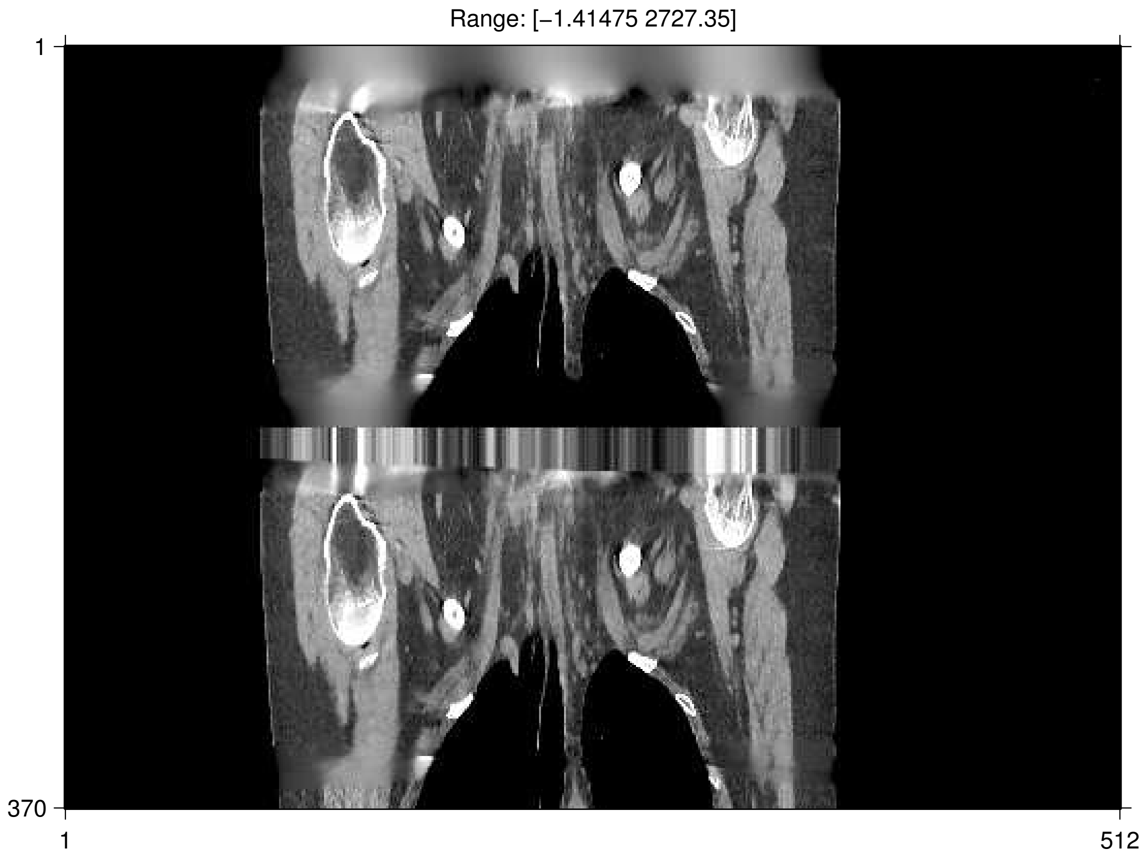


Figure 7.7: Hybrid OS-ICD and ICD reconstructions. *Top:* OS-ICD reconstruction, and *Bottom:* ICD reconstruction. Image window = $[800, 1200]$ HU.

the OS implementation only to make its memory requirements low enough so that it could run on the parallel computer used in this research. There is scope for more software optimization. On the same hardware, software implementations of the same program can have very different compute times depending on the amount of software optimization. Compute time can be reduced by 10 to 100 times by proper software optimization. This should be kept in mind while considering the following results.

The ICD algorithm took 140 minutes and executed 3.9 equits[1]. While the ICD algorithm in the hybrid OS-ICD algorithm took 100 minutes and executed 2.5 equits when

[1]An equit is the short form for equivalent iteration in the ICD algorithm. An equit can be roughly said to be completed if all voxels of the image have been updated.

initialized by the intermediate image. The OS algorithm that created the intermediate image from the FBP image used 12 iterations for a total time of about 215 minutes. Thus, the hybrid OS-ICD algorithm took much longer than the ICD algorithm. This is an unfavorable result for the hybrid OS-ICD algorithm, but it is not discouraging, as many possible methods to accelerate the OS algorithm still exist (Section 7.3.5). As the OS algorithm is accelerated, more iterations can be run in lesser time, and the noise in the intermediate image shall be reduced further. A better intermediate image shall further reduce the time taken for the ICD part of the hybrid OS-ICD algorithm.

The reconstruction time of the parallel OS algorithm is reduced by a factor of about 7 when 8 processor cores are used instead of 1.

7.3.5 Candidate methods to accelerate the OS-algorithm

Various candidate methods to accelerate the OS algorithm are available.

The first is the use of a larger number of subsets. Consider, the *Total time* in the first columns (labeled, *None*) of Table 7.1, 7.2 and Table 7.3. In going from OS 1-subset to OS 41-subset, we get a theoretical convergence rate increase of 41 *i.e.*, need 1 iteration of OS 41-subset instead of 41 iterations of OS 1-subset, with a compute time increase of just 6%. Similarly, in going from OS 41-subset to OS 256-subset, we get a theoretical convergence rate increase of about 6 with a compute time increase of just 36%. These numbers are similar but slightly worse, when regularization is included. But the behavior of the OS 256-subset algorithm is unpredictable in the top and bottom slices. The probable reason for this behavior and possible solutions are suggested in Section 7.3.6.

Another method, as described previously, would be to optimize the software implementation of the OS algorithm.

One of the solutions that reduced the compute time the OS algorithm was the use of the

Generalized-Fair (GF) potential function instead of the q-GGMRF potential function. The GF potential approximates the q-GGMRF potential function closely but does not involve time-consuming function calls to the power function. A large percentage of the compute time of the OS algorithm is attributable to regularization because calls to the gradient and denominator evaluation of the penalty function are nested deeply in nested-loops of the PWLS_OS_SPS algorithm (see Section 6.2.1). Even a small reduction in the time spent in regularization can have a large effect on the overall compute time. From Table 7.2 and Table 7.3, we can see that the total compute time is reduced by a great deal when the q-GGMRF potential function is replaced by the GF potential function.

Another proposed solution is the use of quadratic regularization in the latter OS iterations, and no regularization in the initial OS iterations. If a single global parameter, β, of the quadratic regularization can be found to match the GF potential function over the entire image well, then this method can lead to a large compute time reduction. In Table 7.3, the compute time reduces by nearly 25% for OS 256-subset algorithms, when we replace GF potential function with quadratic. But, such a global β might not exist. To overcome this difficulty, a new algorithm that uses quadratic surrogates in the regularization iterations and is still monotonic in the original cost function is derived in Section 8.2.

	None	Quadratic	GF	q-GGMRF
Total time	13.1 (100%)	13.1 (100%)	13.0 (100%)	13.9 (100%)
Forward projection	6.1 (46.7%)	6.1 (46.8%)	6.2 (47.7%)	6.1 (44.1%)
Back projection	6.9 (53.0%)	6.9 (52.8%)	6.7 (51.6%)	6.9 (49.6%)
Regularization	0.0 (0.0%)	0.1 (0.4%)	0.1 (0.6%)	0.8 (6.1%)
Arithmetic	0.0 (0.3%)	0.0 (0.0%)	0.0 (0.1%)	0.0 (0.1%)

Table 7.1: Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 1 subset and 4 different regularization functions.

7.3.6 Reconstructions from OS-based algorithms in the end slices

OS-based algorithms accelerate convergence by using a subset of the observed data. The image gradient computed by the OS-algorithm is an approximation to the image gradient that would be produced by the entire observed data. In the end-slices of the image volume in the z-direction, the voxels are sampled sparsely due the nature of the helical cone-beam geometry. When the OS-algorithms are used to reconstruct these end-slices, a subset of the already insufficient data is used to compute the image gradient. Thus, the gradient approximation in the end slices gets worse as more and more subsets are employed. This leads to streaking artifacts in the end slices (see Fig. 7.8). We have demonstrated that the OS 41-subset algorithm produces acceptable reconstructions in the end slices, and we can also see from Fig. 7.8 that the OS 256-subset algorithm produces acceptable images in the middle slices. Thus, a possible solution would be to somehow reduce the number of subsets for the end-slices while using a large number of subsets in the middle slices.

	None	Quadratic	GF	q-GGMRF
Total time	13.8 (100%)	15.5 (100%)	17.5 (100%)	58.4 (100%)
Forward projection	6.5 (46.8%)	6.6 (42.5%)	6.5 (37.2%)	6.5 (11.2%)
Back projection	7.1 (51.3%)	7.0 (44.9%)	7.5 (42.8%)	7.1 (12.1%)
Regularization	0.0 (0.0%)	1.6 (10.4%)	3.1 (17.7%)	44.6 (76.3%)
Arithmetic	0.3 (1.9%)	0.3 (2.2%)	0.4 (2.3%)	0.2 (0.4%)

Table 7.2: Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 41 subsets and 4 different regularization functions.

	None	Quadratic	GF	q-GGMRF
Total time	18.8 (100%)	29.0 (100%)	39.6 (100%)	307.5 (100%)
Forward projection	8.3 (44.4%)	8.3 (28.7%)	8.2 (20.6%)	8.3 (2.7%)
Back projection	8.8 (46.7%)	8.6 (29.7%)	9.1 (22.8%)	8.7 (2.8%)
Regularization	0.0 (0.0%)	10.2 (35.4%)	19.9 (50.3%)	288.6 (93.8%)
Arithmetic	1.7 (8.9%)	1.8 (6.2%)	2.5 (6.3%)	2.0 (0.6%)

Table 7.3: Per-iteration compute times (in minutes) and their break-up for OS-based algorithms with 256 subsets and 4 different regularization functions.

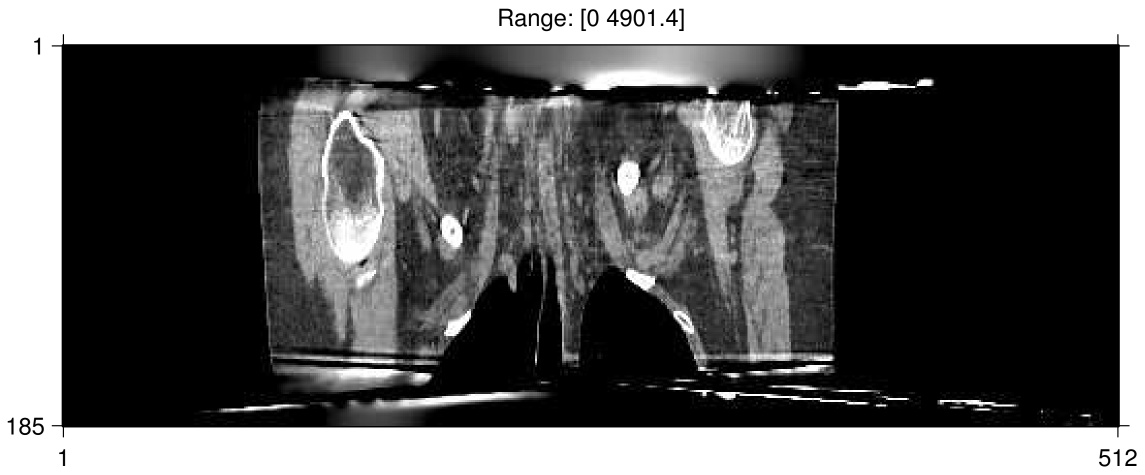


Figure 7.8: A coronal slice of the image reconstruction using the OS 256-subset algorithm.

7.4 Summary

The goal set in this chapter was to lower the compute time of ICD algorithm for practical sized problems. Two approaches, the hybrid OS-ICD and parallel-computation, were tried. The hybrid OS-ICD approach was used to initialize the ICD algorithm with a reconstruction produced by the parallel PWLS_OS_SPS algorithm instead of FBP. Since OS algorithms have a higher convergence rate at low image frequencies than ICD algorithms, the new initialization scheme would benefit the ICD for datasets where the FBP image is very inaccurate, *e.g.*, insufficiently sampled image regions. For the particular dataset investigated here, OS algorithms did not reduce the compute time due to data-insufficiency in the end-slices of helical cone-beam geometry. Possible solutions to this problem were suggested. The compute time of OS algorithms was lowered without degrading the image quality when the q-GGMRF potential function was replaced by the computationally cheaper Generalized-Fair potential function. The parallel version of the PWLS_OS_SPS algorithm was derived. The parallel-computation approach was shown to achieve a reduction in compute time by a large factor in PWLS_OS_SPS algorithms.

7.5 Future work

This section contains three ideas that can possibly lead to image quality improvements. Two of them pertain to OS algorithms, and the third could be used with other algorithms also.

7.5.1 Modification of OS algorithms for helical cone-beam geometry

We found in this chapter, for the helical cone beam geometry, that end slices (ES, Fig. 7.1(a)) are reconstructed well only with OS algorithms that use a low number of subsets, *e.g.*, 41. While the region of interest (ROI, Fig. 7.1(a)) can be reconstructed well with a higher number of subsets, *e.g.*, 256. Thus, the current OS algorithm has to be modified to reconstruct both ROI and ES with a high quality and low reconstruction time. Two ideas are proposed for this purpose: Non-uniform ordered subsets, and alternating minimization between region-of-interest (ROI) and end slices (ES).

Non-uniform ordered subsets

We first divide the full negative log-likelihood into a large number of subsets, say, 256. Now, OS with a large number of subsets can reconstruct ROI well, but it can not reconstruct ES. So, to each subset, we include more rays (or, rays belonging to entire views) that pass through ES. This effectively reduces the number of subsets from the point of view of ES.

The inclusion of more rays is done in such a way that the sum of individual negative log-likelihoods of all subsets yields the full negative log-likelihood. Thus, if a ray i is in M_i subsets then the negative log-likelihood corresponding to that ray is multiplied by $1/M_i$. For example, if a ray is present in two subsets then a factor of 0.5 is multiplied to the two pieces of negative log-likelihood corresponding to that ray.

The disadvantage of this approach is that the total number of views visited for forward and back projection during a given iteration increases by a large number. Thus, each iteration takes longer to execute.

Alternating minimization between region-of-interest (ROI) and end slices (ES)

In this approach, first, ES is held constant and ROI is reconstructed for a few iterations using OS with a large number of subsets. Then, ROI is held constant and ES is reconstructed using OS with a smaller number of subsets. This approach gives the algorithm designer flexibility in choosing the number of iterations of each OS algorithm to execute. This makes the image quality-reconstruction time trade-off easier. In the non-uniform ordered subsets approach, such a trade-off is not possible. This is due to the fixed ratio between the number of iterations at higher subsets and the number of iterations at lower subsets. This is ratio is equal to the ratio of higher number of subsets and the lower number of subsets.

7.5.2 Modification of distance-driven projector for oblique rays

The error in forward and back projections in the distance-driven projector is highest for oblique rays, *i.e.*, those rays that roughly make an angle of 45° with the x-axis. For the oblique rays, imagine the square cartesian grid sitting on top of the original grid that is at 45 degrees to the original grid in the $x - y$ plane. The new cartesian grid has dimensions that are $1/\sqrt{2}$ of the original grid. This grid has sample points from original grid and some new points. These new points are surrounded equidistantly by 4 points of the original grid.

Interpolating for the new grid points should be computationally inexpensive - additions and divide-by-2's. For the oblique rays, the DD projector on the new grid would be costlier by factor of 2. This would be due to the increase in the number of sample points.

In summary, the proposed modification must be able to reduce the error in forward and back projection operations without a high compute time penalty.

CHAPTER 8

Algorithm designs for even more acceleration

This chapter contains three derivations to further accelerate image reconstruction algorithms. In Section 8.1, an algorithm that combines the features of OS and PCG is derived. This algorithm is expected to better the convergence rate of OS algorithms by using a better search direction. In Section 8.2, a relative of the PWLS_OS_SPS algorithm is derived. It aims to reduce the overall compute time by decreasing the number of times computationally expensive functions are computed. Computationally expensive functions are found in the negative log-likelihood part when complex statistical models are employed, and in the regularization part when complex image priors are used. In Section 8.3, we derive a surrogate function that divides the image reconstruction problem into P separate problems among P processors of a computer. The algorithm designer controls the number of times the information is exchanged between the P problems, thus controlling the inter-processor communication.

8.1 OS-PCG hybrid algorithm: OS_PCG_LS

We derive a new algorithm that combines ideas from OS and PCG. OS_PCG_LS algorithm first computes an approximate quadratic surrogate, $\phi_{L,1}(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)})$, at the current iterate, $\boldsymbol{\mu}^{(n)}$. The gradient in this approximate surrogate is the gradient approximation computed using the OS method. This approximate surrogate is thus derived as follows.

$$\begin{aligned}
\phi_L(\boldsymbol{\mu};\boldsymbol{\mu}^{(n)}) &= \sum_{i=1}^{n_d} q_{L,i}(t_i;s_i) \\
&= \sum_{i=1}^{n_d} \dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(n)}]_i)[\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})]_i + \sum_{i=1}^{n_d} \tfrac{1}{2}\breve{c}_i^{(n)}[\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})]_i^2, \text{(from (2.11))} \\
&= (\boldsymbol{G}^T\dot{\boldsymbol{h}}^{(n)})^T(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})^T\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}), \\
&\approx (N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)})^T(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)})^T\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}(\boldsymbol{\mu}-\boldsymbol{\mu}^{(n)}), \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(by OS approximation to the gradient)} \\
&\triangleq \phi_{L,1}(\boldsymbol{\mu};\boldsymbol{\mu}^{(n)}),
\end{aligned}$$

Now, the search direction, $\boldsymbol{d}^{(n)}$, is computed using the PCG procedure outlined in (2.12). The current gradient, $N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)}$, and the previous search direction, $\boldsymbol{d}^{(n-1)}$ are used in (2.12). A line search is performed in this direction on $\phi_{L,1}(\boldsymbol{\mu};\boldsymbol{\mu}^{(n)})$. The line search sets up the cost function $\Phi'(\alpha)$ as follows.

$$\begin{aligned}
\Phi'(\alpha) &\triangleq \phi_{L,1}(\boldsymbol{\mu}^{(n)}+\alpha\boldsymbol{d}^{(n)};\boldsymbol{\mu}^{(n)}), \\
&= (N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)})^T\boldsymbol{d}^{(n)}\alpha + \tfrac{1}{2}\boldsymbol{d}^{(n)^T}\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}\boldsymbol{d}^{(n)}\alpha^2.
\end{aligned}$$

This line search on $\Phi'(\alpha)$ would be very expensive as a forward projection would have to be performed to compute the curvature $\boldsymbol{d}^{(n)^T}\boldsymbol{G}^T\mathcal{D}(\breve{c}_i^{(n)})\boldsymbol{G}\boldsymbol{d}^{(n)}$ at every subiteration (this subiteration refers to iteration over subsets). Using the De Pierro's trick, a curvature that is greater than the previous one is computed. A greater curvature is required in order to create a surrogate that majorizes $\Phi'(\alpha)$:

$$\Phi'(\alpha) \le (N_s\boldsymbol{G}_r^T\dot{\boldsymbol{h}}_r^{(n)})^T\boldsymbol{d}^{(n)}\alpha + \tfrac{1}{2}\boldsymbol{d}^{(n)^T}\mathcal{D}(\sum_{i=1}^{n_d}\breve{c}_i^{(n)}\boldsymbol{G}_i|\boldsymbol{G}_{ij}|)\boldsymbol{d}^{(n)}\alpha^2.$$

$\mathcal{D}(\sum_{i=1}^{n_d}\breve{c}_i^{(n)}\boldsymbol{G}_i|\boldsymbol{G}_{ij}|)$ does not require excessive computation cost as it needs to be computed only once in the WLS case. In the Poisson case, the use of the precomputed curvatures for $\breve{c}_i^{(n)}$ can be explored; precomputed curvatures are also computed once. As we have already used the OS method, high convergence rate is the only advantageous property that our algorithm can possibly possess. That is why the precomputed curvatures would be used in the Poisson case.

Thus, the step size is derived analytically as :

$$\alpha^* = -\frac{(N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})^T \boldsymbol{d}^{(n)}}{\boldsymbol{d}^{(n)T} \mathcal{D}(\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|) \boldsymbol{d}^{(n)}},$$

and, the image update is computed as,

$$\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} - \frac{(N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})^T \boldsymbol{d}^{(n)}}{\boldsymbol{d}^{(n)T} \mathcal{D}(\sum_{i=1}^{n_d} \breve{c}_i^{(n)} \boldsymbol{G}_i |\boldsymbol{G}_{ij}|) \boldsymbol{d}^{(n)}} \boldsymbol{d}^{(n)}. \tag{8.1}$$

Comparing the update (8.1) with (6.3), we find that we are descending in the direction provided by PCG method instead of pixel-wise scaled gradient approximated by the OS method. Note that the direction computed by the PCG method improves the gradient computed by the OS method. Whether the new algorithm improves convergence rate remains to be checked by experiments with real and simulated data.

The pseudo-code is :

```
1.      $\boldsymbol{\mu}^{(0)} = \max(\boldsymbol{\mu}_{FBP}, 0)$
2.      for n=0,1,2,...,$N_1 \times N_s$-1
2.1         Choose subset $r$ = bit_reverse($n \bmod N_1$)
2.2         Compute $\boldsymbol{d}^{(n)}$ by first computing $N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)}$
2.3         Compute $\boldsymbol{\mu}^{(n+1)}$ using (8.1)
2'      end
```

Regularization can be added easily to this algorithm. The total number of equivalent forward and back projections is $2 \times N_1$.

8.2 PWLS_OS_SPS with reduced number of non-linear function evaluations

Regularization involves differencing neighboring voxels and applying a non-linear function to the differences. This non-linear function is called the potential function, ψ, here. Evaluation of the potential function is a computationally expensive operation especially when mathematical operations like power are involved [61]. The PWLS_OS_SPS algorithm described above evaluates the potential function every regularization sub-iteration. Therefore, if N_1, N_s and N_2 are the number of iterations, subsets and regularization sub-iterations respectively, the PWLS_OS_SPS algorithm evaluates the non-linear potential function at every voxel $N_1 \times N_s \times N_2 \times N_n$ times; N_n is the number of neighbors of a voxel considered for regularization ($N_n = 26$ for first-order regularization in 3-dimensions). The algorithm proposed in this section evaluates the non-linear potential function at every voxel $N_1 \times N_n$ times and evaluates the much computationally cheaper quadratic potential function $N_1 \times N_s \times N_2 \times N_n$ times. This saving in compute time comes at the cost of slightly reduced convergence rate. The convergence rate is expected to reduce slightly due to a small increase in curvatures.

As a by-product of the derivation of this method, we shall find a method to adjust the weights sinogram of the negative log-likelihood part as the iterations proceed.

Before the algorithm is derived, a closer look at the representation and notation of the penalty function, $R(\boldsymbol{\mu})$, is required. The penalty function used here is based on [26]. This penalty function has the advantage of providing a way of achieving near-uniform resolution over the image. It is written as :

$$R(\boldsymbol{\mu}) = \sum_{j=1}^{n_p} \sum_{l=1}^{n_p} \mathcal{N}_{j,l} \kappa_j \kappa_l \beta_{j,l} \psi(\mu_j - \mu_l). \tag{8.2}$$

The penalty function is a sum over neighboring voxels of the result of the evaluation of the potential function, ψ, on the voxel difference. $\beta_{j,l}$ is a factor that decides how much weight is given to regularization with respect to the negative log-likelihood. It is also used to change the regularization depending on the relative position and direction of neighboring voxels in the co-ordinate system. Here, $\beta_{j,l}$ depends on $|j-l|$ for neighboring voxels, and so, only 13 unique values of $\beta_{j,l}$ need to be computed and stored. $\boldsymbol{\kappa}$ is an image sized vector that is used to achieve near-uniform resolution over the image (see [26]). $\mathcal{N}_{j,l}$ is an $n_p \times n_p$ size matrix that is used to represent the neighborhood of all voxels. Its elements take one of the two values : 1 or 0. To count each pair of neighboring voxels only once, $\mathcal{N}_{j,l}$ must satisfy : if $\mathcal{N}_{j,l} = 1$, then $\mathcal{N}_{l,j} = 0$. One out of the many values of $\mathcal{N}_{j,l}$ that meets all of the above requirements is :

$$\mathcal{N}_{j,l} = \begin{cases} 1, & l \text{ is in the neighborhood of } j, \text{ and, } l > j \text{ in lexicographic order,} \\ 0, & l \text{ is not in the neighborhood of } j, \text{ or, } l \leq j \text{ in lexicographic order.} \end{cases}$$

This notation of the penalty function is hard to manipulate when computing the gradient and the hessian. A simpler notation is defined as follows. The double-sum of (8.2) is expanded and each term in the expansion is represented by a unique value of the index k. Let K be the total number of terms in the expansion. Thus, each pair of voxels being differenced, *i.e.* (j,l) such that $\mathcal{N}_{j,l} = 1$, has a unique index k. Thus, for each k, define $w_k \overset{\triangle}{=} \kappa_j \kappa_l \beta_{j,l}$. Define a differencing matrix, $\boldsymbol{C}$, of size $K \times n_p$, as follows. $\boldsymbol{C}$ transforms an image into the set of voxel-differences. Each row of $\boldsymbol{C}$ is indexed by k, Within the kth row, all but two terms are non-zero; $\boldsymbol{C}_{k,j} = 1, \boldsymbol{C}_{k,l} = -1$, where (j,l) is the voxel-pair corresponding to k. The penalty function of (8.2) can now be written as

$$R(\boldsymbol{\mu}) = \sum_{k=1}^{K} w_k \psi([\boldsymbol{C}\boldsymbol{\mu}]_k), \tag{8.3}$$

where, k indexes the set $\{(j,l) : \mathcal{N}_{j,l} = 1\}$,

K is the total number of differenced voxel-pairs,

$w_k \overset{\triangle}{=} \kappa_j \kappa_l \beta_{j,l}$.

The proposed algorithm repeatedly creates a quadratic cost function that majorizes the original cost function, $\Phi(\boldsymbol{\mu}) = -L(\boldsymbol{\mu}) + \boldsymbol{R}(\boldsymbol{\mu})$ [1]. The value of the quadratic cost function so created is then reduced using any of the known traditional algorithms. Thus, all evaluations of non-quadratic functions like power, exponentiation, logarithm, *etc.*, are performed in the outermost iteration. In the following section, we first show the creation of the quadratic cost function, and then we show how the OS_SPS algorithm for the PWLS cost function is applied to the computed quadratic cost function. We also show that by making very simple substitutions, the derived algorithm can be used for complex statistical models.

The overall cost function is defined as follows :

$$\begin{aligned}
\Phi(\boldsymbol{\mu}) &= -L(\boldsymbol{\mu}) + R(\boldsymbol{\mu}), \\
-L(\boldsymbol{\mu}) &= \sum_{i=1}^{n_d} h_i([\boldsymbol{G}\boldsymbol{\mu}]_i), \\
R(\boldsymbol{\mu}) &= \sum_{j=1}^{n_p} \sum_{l=1}^{n_p} \mathcal{N}_{j,l} \kappa_j \kappa_l \beta_{j,l} \psi(\mu_j - \mu_l), && \text{(from (8.2))} \\
&= \sum_{k=1}^{K} w_k \psi([\boldsymbol{C}\boldsymbol{\mu}]_k). && \text{(from (8.3))}
\end{aligned}$$

Now, we create a quadratic function that majorizes Φ around the iterate $\boldsymbol{\mu}^{(\tilde{n})}$. The potential

[1] $\Phi(\boldsymbol{\mu})$ is usually defined as $-L(\boldsymbol{\mu}) + \beta\boldsymbol{R}(\boldsymbol{\mu})$. But, $R(\boldsymbol{\mu})$ used here is defined in (8.2), which has β absorbed into it.

function $\psi(t)$ is majorized around a point s using the following inequality :

$$\psi(t) \leq \psi(s) + \dot{\psi}(s)(t-s) + \tfrac{1}{2}\omega_\psi(s)(t-s)^2,$$

where, $\omega_\psi(s) = \dot{\psi}(s)/s$. To majorize $R(\boldsymbol{\mu})$ around $\boldsymbol{\mu}^{(\tilde{n})}$, for each k, set $t = [\boldsymbol{C}\boldsymbol{\mu}]_k$ and $s = [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k$. Thus,

$$\begin{aligned}
\psi([\boldsymbol{C}\boldsymbol{\mu}]_k) &\leq \psi([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k) + \dot{\psi}([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k) \\
&\qquad + \tfrac{1}{2}\omega_\psi([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2 \\
&\overset{\text{opt}}{=} \dot{\psi}([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)[\boldsymbol{C}\boldsymbol{\mu}]_k + \tfrac{1}{2}\omega_\psi([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2. \\
&\qquad\qquad \text{(dropping terms independent of } \boldsymbol{\mu}\text{)}
\end{aligned}$$

The quadratic majorizer for the penalty part can be now written as :

$$\begin{aligned}
\phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(\tilde{n})}) &= \sum_{k=1}^{K} w_k \cdot (\dot{\psi}([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)[\boldsymbol{C}\boldsymbol{\mu}]_k + \tfrac{1}{2}\omega_\psi([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2), \\
&= (\mathcal{D}(w_k)\dot{\boldsymbol{\psi}}(\boldsymbol{\mu}^{(\tilde{n})}))^T \boldsymbol{C}\boldsymbol{\mu} + \frac{1}{2}\sum_{k=1}^{K} w_k \omega_\psi([\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2, \\
&= (\boldsymbol{C}^T \mathcal{D}(w_k)\dot{\boldsymbol{\psi}}(\boldsymbol{\mu}^{(\tilde{n})}))^T \boldsymbol{\mu} \\
&\qquad + \frac{1}{2}\sum_{j=1}^{n_p}\sum_{l=1}^{n_p} \mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})((\mu_j - \mu_l) - (\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})}))^2,
\end{aligned}$$

The storage requirement for $\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})$ in the above surrogate is huge: 13 image volumes. We overcome this problem by creating a new surrogate that requires storage of just 1 image volume as follows.

$$\phi_R(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) \leq (\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \frac{1}{2}\sum_{j=1}^{n_p}\omega_{\psi,j}^{max}\sum_{l=1}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}((\mu_j-\mu_l)-(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})}))^2,$$

$$\text{where, } \omega_{\psi,j}^{max} \triangleq \max_{\{l:\mathcal{N}_{j,l}=1\}}\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})(\text{note: } \omega_\psi(t)>0, \forall t\in\mathbb{R}),$$

$$= (\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \frac{1}{2}\sum_{j=1}^{n_p}\sum_{l=1}^{n_p}\mathcal{N}_{j,l}\omega_{\psi,j}^{max}\kappa_j\kappa_l\beta_{j,l}((\mu_j-\mu_l)-(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})}))^2,$$

$$= (\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{k=1}^{K}\tfrac{1}{2}\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{C}\boldsymbol{\mu}]_k-[\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2,$$

$$\text{where, } \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}) \triangleq \omega_{\psi,j}^{max}\kappa_j\kappa_l\beta_{j,l}, \text{ and } k \text{ indexes the set } \{(j,l):\mathcal{N}_{j,l}=1\}.$$

The weighted least squares (WLS) negative log-likelihood is a cost function that does not involve evaluations of expensive functions like logarithm. But, for complex observation statistics like Poisson, Poisson+Gaussian, Compound-Poisson *etc.*, the negative log-likelihood could involve expensive exponentiation and logarithm operations. In the initial iterations, far from the converged solution, it is conjectured that large benefits in terms of image quality are not obtained by the exact evaluation of these expensive functions. It should be possible to use a quadratic cost function instead of the exact negative log-likelihood in initial iterations. In this section, we derive such a quadratic majorizer to the exact negative log-likelihood. The quadratic majorizer derived here will help test this conjecture.

We majorize $h_i(t)$ using the following inequality :

$$h_i(t) \leq h_i(s) + \dot{h}_i(s)(t-s) + \tfrac{1}{2}\breve{c}_i(s)(t-s)^2,$$

$$\overset{\text{opt}}{=} \dot{h}_i(s)t + \tfrac{1}{2}\breve{c}_i(s)(t-s)^2.$$

(when s is a constant. *e.g.* when s is a function of an iterate like $\boldsymbol{\mu}^{(\tilde{n})}$.)

We majorize $-L(\boldsymbol{\mu})$ around $\boldsymbol{\mu}^{(\tilde{n})}$ by setting for each i, $t = [\boldsymbol{G}\boldsymbol{\mu}]_i$ and $s = [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i$, as follows :

$$\begin{aligned}
\phi_L(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) &= \sum_{i=1}^{n_d} \dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)[\boldsymbol{G}\boldsymbol{\mu}]_i + \tfrac{1}{2}\breve{c}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2,\\
&= \dot{\boldsymbol{h}}(\boldsymbol{\mu}^{(\tilde{n})})^T(\boldsymbol{G}\boldsymbol{\mu}) + \sum_{i=1}^{n_d} \tfrac{1}{2}\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2,\\
&\qquad \text{(where, } \tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})}) \stackrel{\triangle}{=} \breve{c}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i))\\
&= (\boldsymbol{G}^T\dot{\boldsymbol{h}}(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{i=1}^{n_d} \tfrac{1}{2}\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2,\\
&= (\nabla(-L)(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{i=1}^{n_d} \tfrac{1}{2}\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2.
\end{aligned}$$

Computing $\nabla(-L)$ is an computationally expensive operation, but in the initial iterations the computationally cheaper OS approximation to $\nabla(-L)$ should suffice.

Thus, the quadratic majorizer for the objective function for the general negative log-likelihood is :

$$\begin{aligned}
\tilde{\Phi}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) &= \phi_L(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) + \phi_R(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}),\\
&= (\nabla - L(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{i=1}^{n_d} \tfrac{1}{2}\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2\\
&\quad + (\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{k=1}^{K} \tfrac{1}{2}\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2,\\
&= (\nabla\Phi(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu} + \sum_{i=1}^{n_d} \tfrac{1}{2}\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{G}\boldsymbol{\mu}]_i - [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)^2\\
&\quad + \sum_{k=1}^{K} \tfrac{1}{2}\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2. \;(\because \Phi(\boldsymbol{\mu}) = -L(\boldsymbol{\mu}) + R(\boldsymbol{\mu}))
\end{aligned}$$

Note here that the weights sinogram $\tilde{\boldsymbol{w}}_L$ is being adjusted as the iterations proceed. The quadratic majorizer for the objective function for the WLS negative log-likelihood is :

$$
\begin{aligned}
\tilde{\Phi}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) &= -L(\boldsymbol{\mu}) + \phi_R(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}), \\
&= \sum_{i=1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G}\boldsymbol{\mu}]_i - l_i)^2 + (\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T \boldsymbol{\mu} \\
&\quad + \sum_{k=1}^{K} \tfrac{1}{2} \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2.
\end{aligned} \tag{8.4}
$$

There are many similarities in both the constructions of $\tilde{\Phi}$. By exploiting these similarities, the algorithms used to minimize $\tilde{\Phi}$ for the WLS case can be reused for the more general case. If we replaced $\nabla R(\boldsymbol{\mu}^{(\tilde{n})})$ by $\nabla \Phi(\boldsymbol{\mu}^{(\tilde{n})})$, l_i by $[\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i$, and w_i by $\tilde{w}_{L,i}(\boldsymbol{\mu}^{(\tilde{n})})$, we convert $\tilde{\Phi}$ from the WLS case to the general case. These replacements are valid because the quantities changed are constants in the loop over $\tilde{n}$. These similarities also imply that we have changed the general image reconstruction problem into a sequence of QPWLS problems.

The skeleton of the proposed algorithm is shown in Table 8.1. This concludes the first part of the derivation.

```
Initialize $\boldsymbol{\mu}^{(\tilde{n})} = \boldsymbol{\mu}_{initial}$
loop over $\tilde{n}$
    $\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \tilde{\Phi}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})})$          (see (8.4))
    $\boldsymbol{\mu}^{(\tilde{n})} = \boldsymbol{\mu}^*$
end loop
```

Table 8.1: Skeleton of algorithm for reducing non-linear function evaluations

We now apply the framework derived above to the PWLS cost function. We use the OS_SPS algorithm to perform the minimization shown in Table 8.1. The following algorithm uses the same structure as the algorithm from Section 6.2.2. There are two levels of surrogates implemented using an outer loop (indexed by n) and an inner loop (indexed by m). In the outer loop, the first surrogate is created as a quadratic function by majorizing the negative log-likelihood part. In the inner loop, the second surrogate is also a quadratic cost function that majorizes the penalty part.

First, let us apply the ordered-subsets idea and DePierro's trick to the quadratic cost function majorizing the likelihood part, as is done in Section 6.2.1 :

$$\begin{aligned}
&\sum_{i=1}^{n_d} \tfrac{1}{2} w_i([\boldsymbol{G}\boldsymbol{\mu}]_i - l_i)^2 \\
&= \sum_{i=1}^{n_d} h_i([\boldsymbol{G}\boldsymbol{\mu}]_i), \qquad \text{(define here, } h_i(t) = \tfrac{1}{2} w_i (t - l_i)^2\text{)}, \\
&\approx (N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T \boldsymbol{G}^T \mathcal{D}(w_i) \boldsymbol{G} (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}), \\
&\qquad \text{(OS approximation to the gradient)} \\
&\leq (N_s \boldsymbol{G}_r^T \dot{\boldsymbol{h}}_r^{(n)})^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T \mathcal{D}(\sum_{i=1}^{n_d} w_i \boldsymbol{G}_i |\boldsymbol{G}_{ij}|)(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}), \\
&\qquad \text{(by DePierro's trick and } \boldsymbol{G}_i \triangleq \sum_{j=1}^{n_p} |\boldsymbol{G}_{ij}|\text{)}, \\
&= (N_s \boldsymbol{G}_r^T \mathcal{D}(\boldsymbol{w}_r)(\boldsymbol{G}_r \boldsymbol{\mu}^{(n)} - \boldsymbol{l}_r))^T (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T \mathcal{D}_L (\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}), \\
&\qquad \text{(where, } \mathcal{D}_L \triangleq \mathcal{D}(\sum_{i=1}^{n_d} w_i \boldsymbol{G}_i |\boldsymbol{G}_{ij}|)\text{)}
\end{aligned}$$

Before going to the surrogate for the regularization part, let us prove the following property :

$$\begin{aligned}
&([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2 \\
&= ([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k + [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2, \\
&= ([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)^2 + ([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2 \\
&\qquad + 2([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k), \\
&\overset{\text{opt}}{=} 2 \cdot \Big(([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k) + \tfrac{1}{2}([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)^2\Big).
\end{aligned}$$

The surrogate for the penalty part can now be written as :

$$\sum_{k=1}^{K} \tfrac{1}{2}\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)^2$$

$$\overset{\text{opt}}{=} \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})\Big(([\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(\tilde{n})}]_k)([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k) + \tfrac{1}{2}([\boldsymbol{C}\boldsymbol{\mu}]_k - [\boldsymbol{C}\boldsymbol{\mu}^{(n,m)}]_k)^2\Big),$$

$$= (\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}))\boldsymbol{C}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{C}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})$$

$$+ \tfrac{1}{2}(\boldsymbol{C}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}))^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})(\boldsymbol{C}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})),$$

$$= (\boldsymbol{C}^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}))\boldsymbol{C}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(\tilde{n})}))^T(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})$$

$$+ \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T\{\boldsymbol{C}^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})\boldsymbol{C}\}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}),$$

$$\leq (\boldsymbol{C}^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}))\boldsymbol{C}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(\tilde{n})}))^T(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T\mathcal{D}_R^{(\tilde{n})}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})$$

(by DePierro's trick, $\boldsymbol{C}^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}))\boldsymbol{C} \prec \mathcal{D}_R^{(\tilde{n})} \triangleq \mathcal{D}(\sum_{k=1}^{n_d} \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})\boldsymbol{C}_k|\boldsymbol{C}_{kj}|)$,

$\boldsymbol{C}_k \triangleq \sum_{j=1}^{n_p} |\boldsymbol{C}_{kj}|)$.

Thus, the innermost quadratic cost function is :

$$(\nabla R(\boldsymbol{\mu}^{(\tilde{n})}))^T\boldsymbol{\mu}$$

$$+ (g_L^{(n)})^T(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})^T\mathcal{D}_L(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)})$$

$$+ (g_R^{(n,m)})^T(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}) + \tfrac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)})^T\mathcal{D}_R^{(\tilde{n})}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}),$$

where, $g_L^{(n)} \triangleq N_s\boldsymbol{G}_r^T\mathcal{D}(\boldsymbol{w}_r)(\boldsymbol{G}_r\boldsymbol{\mu}^{(n)} - \boldsymbol{l}_r), g_R^{(n,m)} \triangleq \boldsymbol{C}^T\mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}))\boldsymbol{C}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(\tilde{n})})$.

The first derivative of the cost function is :

$$\nabla R(\boldsymbol{\mu}^{(\tilde{n})}) + g_L^{(n)} + \mathcal{D}_L(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n)}) + g_R^{(n,m)} + \mathcal{D}_R^{(\tilde{n})}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(n,m)}),$$

$$= (\mathcal{D}_L + \mathcal{D}_R^{(\tilde{n})})\boldsymbol{\mu} - (\mathcal{D}_L\boldsymbol{\mu}^{(n)} - g_L^{(n)}) - (\mathcal{D}_R^{(\tilde{n})}\boldsymbol{\mu}^{(n,m)} - \nabla R(\boldsymbol{\mu}^{(\tilde{n})}) - g_R^{(n,m)}).$$

Setting the first derivative to 0, we get the image update as :

$$\boldsymbol{\mu}^{(n,m+1)} = [\frac{(\mathcal{D}_R^{(\tilde{n})}\boldsymbol{\mu}^{(n,m)} - g_R^{(n,m)}) + (\mathcal{D}_L\boldsymbol{\mu}^{(n)} - g_L^{(n)}) - \nabla R(\boldsymbol{\mu}^{(\tilde{n})})}{\mathcal{D}_L + \mathcal{D}_R^{(\tilde{n})}}]_+.$$

The final overall algorithm is shown in Table 8.2. The implementation of the new algorithm needs three more image volumes than the traditional PWLS_OS_SPS algorithm of Section 6.2.2 : $\boldsymbol{\mu}^{(\tilde{n})}$, $\boldsymbol{\omega}_{\boldsymbol{\psi}}^{\boldsymbol{max}}$, and $\nabla R(\boldsymbol{\mu}^{(\tilde{n})})$. When OS is employed and this algorithm is used where complex statistics are involved, one extra forward and back projection is involved in the loop over $\tilde{n}$. The extra projection operations will probably not be required when CG methods are employed. Because, simplification due to the full forward projection result being used twice would be possible. Another interesting feature of this method is that all division operations can also be performed in the outermost loop. In traditional PWLS_OS_SPS with non-quadratic regularization, division operations are required in the innermost loop.

```
Initialize $\boldsymbol{\mu}^{(\tilde{n})} = \boldsymbol{\mu}_{initial}$
loop over $\tilde{n}$
    Compute $\forall j, \omega_{\psi,j}^{max} = \max_{\{l:\mathcal{N}_{j,l}=1\}} \omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})$
    Note $\forall k, \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}) \triangleq \omega_{\psi,j}^{max} \kappa_j \kappa_l \beta_{j,l},\ k$ indexes $\{(j,l) : \mathcal{N}_{j,l} = 1\}$
    Compute $\mathcal{D}_R^{(\tilde{n})} = \mathcal{D}(\sum_{k=1}^{n_d} \tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})}) \boldsymbol{C}_k |\boldsymbol{C}_{kj}|)$
    Compute $\mathcal{D}_L = \mathcal{D}(\sum_{i=1}^{n_d} w_i \boldsymbol{G}_i |\boldsymbol{G}_{ij}|)$
    Compute $\nabla R(\boldsymbol{\mu}^{(\tilde{n})})$
    Save $\boldsymbol{\mu}^{(\tilde{n})}$
    Initialize $\boldsymbol{\mu}^{(n)} = \boldsymbol{\mu}^{(\tilde{n})}$
    loop over $n$
        Compute subset $r$ = bit_reverse($n \bmod N_s$)
        Compute $z_L^{(n)} = \mathcal{D}_L \boldsymbol{\mu}^{(n)} - N_s \boldsymbol{G}_r^T \mathcal{D}(\boldsymbol{w}_r)(\boldsymbol{G}_r \boldsymbol{\mu}^{(n)} - \boldsymbol{l}_r)$
        Initialize $\boldsymbol{\mu}^{(n,m)} = \boldsymbol{\mu}^{(n)}$
        loop over $m$
            $g_R^{(n,m)} = \boldsymbol{C}^T \mathcal{D}(\tilde{w}_k(\boldsymbol{\mu}^{(\tilde{n})})) \boldsymbol{C}(\boldsymbol{\mu}^{(n,m)} - \boldsymbol{\mu}^{(\tilde{n})})$

            $$\boldsymbol{\mu}^{(n,m+1)} = [\frac{(\mathcal{D}_R^{(\tilde{n})} \boldsymbol{\mu}^{(n,m)} - g_R^{(n,m)}) + z_L^{(n)} - \nabla R(\boldsymbol{\mu}^{(\tilde{n})})}{\mathcal{D}_L + \mathcal{D}_R^{(\tilde{n})}}]_+.$$

        end loop over $m$
        Update $\boldsymbol{\mu}^{(n)}$ with last value of $\boldsymbol{\mu}^{(n,m)}$
    end loop over $n$
    Update $\boldsymbol{\mu}^{(\tilde{n})}$ with last value of $\boldsymbol{\mu}^{(n)}$
end loop over $\tilde{n}$
$\boldsymbol{\mu}^{(\tilde{n})}$ is the output image
```

Table 8.2: Algorithm for reducing non-linear function evaluations

8.3 Surrogate cost function for reduced inter-processor communication

Inter-processor communication is a potential bottleneck when image reconstruction will be implemented on computers with a large number of processor cores. Advancements in computing speed in the near future are likely to happen by addition of more and more cores rather than by an increase in clock speed. The solution to this problem proposed in this section, is to first divide the reconstructed image volume among the Memory and Processor Units (MPUs) of the computer[2]. A new surrogate cost function that is a sum of cost functions, each of which individually depends on the image volume segment alloted to a MPU is then derived. If a sinogram-bin in the negative log-likelihood or a voxel-difference in the penalty depends solely on the image volume for that MPU then the corresponding term from the original cost function is retained. If that is not the case, then the corresponding term from the original cost function is replaced by a sum of quadratic surrogates. Now, each MPU produces a new iterate of its image volume segment by optimizing its cost function in parallel with other MPUs. Next, the MPUs exchange the latest iterates of the image volume segments from neighboring MPUs. This process is repeated over and over until convergence criteria are met.

The cost function derived here is a true surrogate of the original cost function, and so, the value of the original cost function is guaranteed to reduce every iteration. Also, the cost function derived here is independent of the algorithm being employed. In other words, OS, CG and ICD algorithms can be used within the framework introduced here. The cost function derived here can also be easily used for targeted reconstruction (see Section 8.3.1). While the cost function has the potential of providing big advantages, it

[2]An MPU is defined here as a core or a group of cores along with fast accessible local random-access memory.

also produces some overhead (see Section 8.3.2). Experiments are needed to check that the overhead does not outweigh the benefits.

First, let us tackle the negative log-likelihood part. The techniques used for likelihood would be employed again for the penalty part. Consider, the negative log-likelihood part of the original cost function :

$$-L(\boldsymbol{\mu}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{G}\boldsymbol{\mu}]_i).$$

Let us divide the image volume $\boldsymbol{\mu}$ into P disjoint segments, $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_P \end{bmatrix}$ (see Fig. 7.1 and Fig. 7.2). If a ray i lies solely in an image volume segment p, then the corresponding term in the negative log-likelihood, h_i, is retained without modification. Now, let $\mathcal{P}_i$ be the set of image volume segments through which the ray i passes. Therefore, for ray i that passes through a single image volume segment p, $\mathcal{P}_i = \{p\}$, a quadratic surrogate over h_i is not created. Corresponding to the image volume segment division defined above, the system model $\boldsymbol{G}$ can be written as a concatenation of operators as follows : $\boldsymbol{G} = [\boldsymbol{G}_1 \dots \boldsymbol{G}_P]$. Thus, $[\boldsymbol{G}\boldsymbol{\mu}]_i = [\sum_{p=1}^{P} \boldsymbol{G}_p\boldsymbol{\mu}_p]_i = [\sum_{p\in\mathcal{P}_i} \boldsymbol{G}_p\boldsymbol{\mu}_p]_i = \sum_{p\in\mathcal{P}_i} [\boldsymbol{G}_p\boldsymbol{\mu}_p]_i$. For a ray i that passes through more than one image volume segment, we can write the quadratic surrogate for it at surrogate point $\boldsymbol{\mu}^{(\tilde{n})}$ as :

$$\begin{aligned}
h_i([\boldsymbol{G\mu}]_i) &\leq \dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)([\boldsymbol{G\mu}]_i - [\boldsymbol{G\mu}^{(\tilde{n})}]_i) + \tfrac{1}{2}\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)([\boldsymbol{G\mu}]_i - [\boldsymbol{G\mu}^{(\tilde{n})}]_i)^2, \\
&\overset{\text{opt}}{=} \dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)\sum_{p\in\mathcal{P}_i}[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)(\sum_{p\in\mathcal{P}_i}[\boldsymbol{G}_p(\boldsymbol{\mu}_p - \boldsymbol{\mu}_p^{(\tilde{n})})]_i)^2, \\
&= \dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)\sum_{p\in\mathcal{P}_i}[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)(\sum_{p\in\mathcal{P}_i}\alpha_{i,p}\frac{1}{\alpha_{i,p}}[\boldsymbol{G}_p(\boldsymbol{\mu}_p - \boldsymbol{\mu}_p^{(\tilde{n})})]_i)^2, \\
&\qquad (0 < \alpha_{i,p} < 1, \sum_{p\in\mathcal{P}_i}\alpha_{i,p} = 1), \\
&\leq \dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)\sum_{p\in\mathcal{P}_i}[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)\sum_{p\in\mathcal{P}_i}\alpha_{i,p}(\frac{1}{\alpha_{i,p}}[\boldsymbol{G}_p(\boldsymbol{\mu}_p - \boldsymbol{\mu}_p^{(\tilde{n})})]_i)^2, \\
&= \sum_{p\in\mathcal{P}_i}\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}[\boldsymbol{G}_p(\boldsymbol{\mu}_p - \boldsymbol{\mu}_p^{(\tilde{n})})]_i^2, \\
&\overset{\text{opt}}{=} \sum_{p\in\mathcal{P}_i}\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i^2 - 2[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i), \\
&= \sum_{p\in\mathcal{P}_i}(\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i) - \frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i)[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i + \tfrac{1}{2}\frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i^2, \\
&= \sum_{p\in\mathcal{P}_i}\tfrac{1}{2}\frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}\Big([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i^2 - 2([\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i - \alpha_{i,p}\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)})[\boldsymbol{G}_p\boldsymbol{\mu}_p]_i\Big), \\
&\overset{\text{opt}}{=} \sum_{p\in\mathcal{P}_i}\tfrac{1}{2}\frac{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{\alpha_{i,p}}\Big([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i - ([\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i - \alpha_{i,p}\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)})\Big)^2.
\end{aligned}$$

This cost function suggests that the line integral for the estimation of $\boldsymbol{\mu}_p$ using ray i is $[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i - \alpha_{i,p}\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}$. We choose $\alpha_{i,p} = \frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i} = \frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}{\sum_{p\in\mathcal{P}_i}[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}$ (if $[\boldsymbol{G\mu}^{(\tilde{n})}]_i = 0$, then we can use $\alpha_{i,p} = 1/card(\mathcal{P}_i)$). This value of $\alpha_{i,p}$ satisfies the convexity property, and it gives us an intuitively satisfying result in the WLS case. In the WLS case, $h_i(t) = \frac{1}{2}w_i(t - l_i)^2$, and $\dot{h}_i(t) = w_i(t - l_i)$ and $\breve{c}_i(t) = w_i$. Thus,

$$[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i - \alpha_{i,p}\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{\breve{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}) = [\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i - \frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i}\frac{w_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i - l_i)}{w_i}) = \frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i}l_i.$$

The line integral observed for ray i is l_i, and the fraction of l_i due to $\boldsymbol{\mu}_p$ is $\frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{true}]_i}{[\boldsymbol{G\mu}^{true}]_i}l_i$. Note that the line-integral derived above for estimation of $\boldsymbol{\mu}_p$ is $\frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i}l_i$, which is analogous to $\frac{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{true}]_i}{[\boldsymbol{G\mu}^{true}]_i}l_i$. The curvature is correspondingly increased by a fraction $\frac{[\boldsymbol{G\mu}^{(\tilde{n})}]_i}{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}$.

Thus, the surrogate for the negative log-likelihood can now be written as :

$$-\tilde{L}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) = \sum_{i=1}^{n_d} \tilde{h}_i([\boldsymbol{G\mu}]_i;[\boldsymbol{G\mu}^{(\tilde{n})}]_i),$$

where, $\tilde{h}_i([\boldsymbol{G\mu}]_i;[\boldsymbol{G\mu}^{(\tilde{n})}]_i) \triangleq$

$$\begin{cases} h_i([\boldsymbol{G\mu}]_i), & card(\mathcal{P}_i)=1, \\ \sum_{p\in\mathcal{P}_i} \frac{1}{2}\frac{[\boldsymbol{G\mu}^{(\tilde{n})}]_i \check{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}\Big([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i - [\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i(1-\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i\check{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)})\Big)^2, & card(\mathcal{P}_i)>1. \end{cases}$$

We move the sum over p into the cost function to find :

$$-\tilde{L}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) = \sum_{i=1}^{n_d}\sum_{p\in\mathcal{P}_i} \tilde{h}_{i,p}([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i;[\boldsymbol{G\mu}^{(\tilde{n})}]_i),$$

where, $\tilde{h}_{i,p}([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i;[\boldsymbol{G\mu}^{(\tilde{n})}]_i) \triangleq$

$$\begin{cases} h_i([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i), & card(\mathcal{P}_i)=1, \\ \frac{1}{2}\frac{[\boldsymbol{G\mu}^{(\tilde{n})}]_i \check{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)}{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}\Big([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i - [\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i(1-\frac{\dot{h}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i}{[\boldsymbol{G\mu}^{(\tilde{n})}]_i\check{c}_i([\boldsymbol{G\mu}^{(\tilde{n})}]_i)})\Big)^2, & card(\mathcal{P}_i)>1. \end{cases}$$

We can rearrange the terms of the double sum to the following form :

$$-\tilde{L}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})}) = \sum_{p=1}^{P} -\tilde{L}_p(\boldsymbol{\mu}_p;\boldsymbol{\mu}^{(\tilde{n})}),$$

$$-\tilde{L}_p(\boldsymbol{\mu}_p;\boldsymbol{\mu}^{(\tilde{n})}) \triangleq \sum_{i\in\mathcal{I}_p} \tilde{h}_{i,p}([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i;[\boldsymbol{G\mu}^{(\tilde{n})}]_i)$$

where, $\mathcal{I}_p = \{i : p \in \mathcal{P}_i\}$ is the set of rays passing through image volume segment p, and $-\tilde{L}_p(\boldsymbol{\mu}_p;\boldsymbol{\mu}^{(\tilde{n})})$ is the surrogate of the negative log-likelihood for the pth image volume.

Now, for the regularization part. Recall from (8.2), the definition of the penalty function implemented here as :

$$R(\boldsymbol{\mu}) = \sum_{j=1}^{n_p}\sum_{l=1}^{n_p} \mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\psi(\mu_j-\mu_l).$$

Once the image volume is split into P components as $\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_1\\ \vdots \\ \boldsymbol{\mu}_P\end{bmatrix}$, the penalty function can be written as :

$$R(\boldsymbol{\mu}) = \sum_{p=1}^{P} R_p(\boldsymbol{\mu}),$$
$$R_p(\boldsymbol{\mu}) \triangleq \sum_{j:\mu_j \in \boldsymbol{\mu}_p} \sum_{l=1}^{n_p} \mathcal{N}_{j,l} \kappa_j \kappa_l \beta_{j,l} \psi(\mu_j - \mu_l).$$

Our goal here is to create a surrogate function $\tilde{R}_p$ over $R_p(\boldsymbol{\mu})$ such that $\tilde{R}_p$ has $\boldsymbol{\mu}_p$ as its independent variable. Let us consider the specific case, where the division of $\boldsymbol{\mu}$ into image volume segments $\boldsymbol{\mu}_p$ is done by defining $\boldsymbol{\mu}_p$ as a bunch of contiguous slices. In this case, we find that for voxel-differences that solely depend on the voxels of $\boldsymbol{\mu}_p$ (μ_j and μ_l are both elements of $\boldsymbol{\mu}_p$), the corresponding term in R_p can be retained without changes. We also note that at the top and bottom slices of $\boldsymbol{\mu}_p$, voxel differences are taken between $\boldsymbol{\mu}_p$ and its neighboring image volume segments. For voxel-differences that straddle neighboring image volume segments $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$, we aim to replace the corresponding terms in R_p and R_q with quadratic surrogates.

Consider, the quadratic surrogate of $\psi(\mu_j - \mu_l)$ at $\boldsymbol{\mu}^{(\tilde{n})}$:

$$\tilde{\psi}(\mu_j - \mu_l; \mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})$$

$$= \dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\overline{\mu_j - \mu_l} - \overline{\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})}}) + \tfrac{1}{2}\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\overline{\mu_j - \mu_l} - \overline{\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})}})^2,$$

$$\le \dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})}) + \tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})})^2$$

$$+ \dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(-\mu_l + \mu_l^{(\tilde{n})}) + \tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(-\mu_l + \mu_l^{(\tilde{n})})^2$$

(using, $(a+b)^2 \le 2(a^2+b^2), a, b \in \mathbb{R}$),

$$= \dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})}) + \tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})})^2$$

$$+ \dot{\psi}(\mu_l^{(\tilde{n})} - \mu_j^{(\tilde{n})})(\mu_l - \mu_l^{(\tilde{n})}) + \tfrac{1}{2}2\omega_\psi(\mu_l^{(\tilde{n})} - \mu_j^{(\tilde{n})})(\mu_l - \mu_l^{(\tilde{n})})^2$$

(since, $\dot{\psi}(-t) = -\dot{\psi}(t), \omega_\psi(-t) = \omega_\psi(t)$),

$$\overset{\text{opt}}{=} \tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})} + \frac{\dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})})^2$$

$$+ \tfrac{1}{2}2\omega_\psi(\mu_l^{(\tilde{n})} - \mu_j^{(\tilde{n})})(\mu_l - \mu_l^{(\tilde{n})} + \frac{\dot{\psi}(\mu_l^{(\tilde{n})} - \mu_j^{(\tilde{n})})}{2\omega_\psi(\mu_l^{(\tilde{n})} - \mu_j^{(\tilde{n})})})^2$$

(using, $ax + \tfrac{1}{2}bx^2 \overset{\text{opt}}{=} \tfrac{1}{2}b(x + \dfrac{a}{b})^2$).

An interesting observation can be made when the potential function is quadratic, $\psi(t) = \frac{1}{2}t^2$. Here, $\dot{\psi}(t) = t, \omega_\psi(t) = 1$, and so, $\tilde{\psi}(\mu_j - \mu_l; \mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})}) = \frac{1}{2}2(\mu_j - \frac{\mu_j^{(\tilde{n})}+\mu_l^{(\tilde{n})}}{2})^2 + \frac{1}{2}2(\mu_l - \frac{\mu_l^{(\tilde{n})}+\mu_j^{(\tilde{n})}}{2})^2$. This is an intuitively satisfying result which shows that when the penalty function $\frac{1}{2}(\mu_j - \mu_l)^2$ is decoupled into a sum of two quadratic surrogate functions that depend on μ_j and μ_l individually, the resulting penalty function tries to bring μ_j and μ_l closer to the average of old iterate values : $\frac{\mu_l^{(\tilde{n})}+\mu_j^{(\tilde{n})}}{2}$. The resulting cost function also prevents the algorithm from taking large steps in the direction of average of old iterates by doubling the curvatures.

The surrogate of $R(\boldsymbol{\mu})$ can now be written as :

$$
\begin{aligned}
&\tilde{R}(\boldsymbol{\mu};\boldsymbol{\mu}^{(\tilde{n})})\\
&=\sum_{p=1}^{P}\sum_{j:\mu_j\in\boldsymbol{\mu}_p}\Big(\sum_{l=1,\mu_l\in\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\psi(\mu_j-\mu_l)\\
&\qquad\qquad+\sum_{l=1,\mu_l\notin\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\tilde{\psi}(\mu_j-\mu_l;\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})\Big),\\
&=\sum_{p=1}^{P}\sum_{j:\mu_j\in\boldsymbol{\mu}_p}\Big(\sum_{l=1,\mu_l\in\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\psi(\mu_j-\mu_l)\\
&\quad+\sum_{l=1,\mu_l\notin\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\Big(\tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})(\mu_j-\mu_j^{(\tilde{n})}+\frac{\dot{\psi}(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})})^2\\
&\quad+\tfrac{1}{2}2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})(\mu_l-\mu_l^{(\tilde{n})}+\frac{\dot{\psi}(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})}{2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})})^2\Big)\Big),\\
&=\sum_{p=1}^{P}\sum_{j:\mu_j\in\boldsymbol{\mu}_p}\Big(\sum_{l=1,\mu_l\in\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\psi(\mu_j-\mu_l)\\
&\quad+\sum_{l=1,\mu_l\notin\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})(\mu_j-\mu_j^{(\tilde{n})}+\frac{\dot{\psi}(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})})^2\\
&\quad+\sum_{l=1,\mu_l\notin\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\tfrac{1}{2}2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})(\mu_l-\mu_l^{(\tilde{n})}+\frac{\dot{\psi}(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})}{2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})})^2\Big),\\
&=\sum_{p=1}^{P}\sum_{j:\mu_j\in\boldsymbol{\mu}_p}\Big(\sum_{l=1}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\tilde{\psi}_{j,l}\\
&\quad+\sum_{l=1,\mu_l\notin\boldsymbol{\mu}_p}^{n_p}\mathcal{N}_{j,l}\kappa_j\kappa_l\beta_{j,l}\tfrac{1}{2}2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})(\mu_l-\mu_l^{(\tilde{n})}+\frac{\dot{\psi}(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})}{2\omega_\psi(\mu_l^{(\tilde{n})}-\mu_j^{(\tilde{n})})})^2\Big),\\
&\text{where, }\tilde{\psi}_{j,l}\triangleq\begin{cases}\psi(\mu_j-\mu_l), & \mu_l\in\boldsymbol{\mu}_p,\\ \tfrac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})(\mu_j-\mu_j^{(\tilde{n})}+\frac{\dot{\psi}(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})})^2, & \mu_l\notin\boldsymbol{\mu}_p.\end{cases}
\end{aligned}
$$

The quadratic surrogate derived over the original cost function, ψ, has two quadratic terms. This has caused generation of extra terms along the boundaries of two neighboring image volume segments. All of these extra terms can be neatly included into the original form if we expand the definition of $\mathcal{N}_{j,l}$ as follows :

$$\tilde{\mathcal{N}}_{j,l} \stackrel{\triangle}{=} \begin{cases} \mathcal{N}_{j,l}, & \mu_j, \mu_l \text{ in the same image volume segment,} \\ 1, & \mu_j, \mu_l \text{ in different image volume segments, and, } \mathcal{N}_{j,l} = 1 \text{ or } \mathcal{N}_{l,j} = 1, \\ 0, & \mu_j, \mu_l \text{ in different image volume segments, and, } \mathcal{N}_{j,l} = 0 \text{ and } \mathcal{N}_{l,j} = 0. \end{cases}$$

A necessary condition that has to be satisfied here is $\beta_{j,l} = \beta_{l,j}$. All penalty functions considered in this thesis satisfy this condition.

Thus,

$$\tilde{R}(\boldsymbol{\mu}; \boldsymbol{\mu}^{(\tilde{n})}) = \sum_{p=1}^{P} \sum_{j:\mu_j \in \boldsymbol{\mu}_p} \sum_{l=1}^{n_p} \tilde{\mathcal{N}}_{j,l} \kappa_j \kappa_l \beta_{j,l} \tilde{\psi}_{j,l} = \sum_{p=1}^{P} R_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}),$$

$$R_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}) \stackrel{\triangle}{=} \sum_{j:\mu_j \in \boldsymbol{\mu}_p} \sum_{l=1}^{n_p} \tilde{\mathcal{N}}_{j,l} \kappa_j \kappa_l \beta_{j,l} \tilde{\psi}_{j,l},$$

$$\text{where, } \tilde{\psi}_{j,l} = \begin{cases} \psi(\mu_j - \mu_l), & \mu_l \in \boldsymbol{\mu}_p, \\ \frac{1}{2} 2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})(\mu_j - \mu_j^{(\tilde{n})} + \frac{\dot{\psi}(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})} - \mu_l^{(\tilde{n})})})^2, & \mu_l \notin \boldsymbol{\mu}_p. \end{cases}$$

Thus, the surrogate over the overall cost function is

$$\tilde{\Phi}(\boldsymbol{\mu}; \boldsymbol{\mu}^{(\tilde{n})}) = \sum_{p=1}^{P} \tilde{\Phi}_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}),$$

$$\tilde{\Phi}_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}) \stackrel{\triangle}{=} -\tilde{L}_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}) + \tilde{R}_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}),$$

$$-\tilde{L}_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}) = \sum_{i \in \mathcal{I}_p} \tilde{h}_{i,p}([\boldsymbol{G}_p \boldsymbol{\mu}_p]_i; [\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i),$$

$$R_p(\boldsymbol{\mu}_p; \boldsymbol{\mu}^{(\tilde{n})}) = \sum_{j:\mu_j \in \boldsymbol{\mu}_p} \sum_{l=1}^{n_p} \tilde{\mathcal{N}}_{j,l} \kappa_j \kappa_l \beta_{j,l} \tilde{\psi}_{j,l},$$

$$\tilde{h}_{i,p}([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i;[\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i) =$$
$$\begin{cases} h_i([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i), & card(\mathcal{P}_i)=1, \\ \frac{1}{2}\frac{[\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i\check{c}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)}{[\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i}\Big([\boldsymbol{G}_p\boldsymbol{\mu}_p]_i - [\boldsymbol{G}_p\boldsymbol{\mu}_p^{(\tilde{n})}]_i(1-\frac{\dot{h}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i}{[\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i\check{c}_i([\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i)})\Big)^2, & card(\mathcal{P}_i)>1, \end{cases}$$

$$\tilde{\psi}_{j,l} = \begin{cases} \psi(\mu_j-\mu_l), & \mu_l \in \boldsymbol{\mu}_p, \\ \frac{1}{2}2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})(\mu_j-\mu_j^{(\tilde{n})}+\frac{\dot{\psi}(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})}{2\omega_\psi(\mu_j^{(\tilde{n})}-\mu_l^{(\tilde{n})})})^2, & \mu_l \notin \boldsymbol{\mu}_p. \end{cases}$$

The skeleton of a parallel algorithm that uses this quadratic surrogate is shown in Table 8.3.

```
Initialize $\boldsymbol{\mu}^{(\tilde{n})} = \boldsymbol{\mu}_{initial}$
loop over $\tilde{n}$
    loop over p (execute in parallel)
        Compute parameters of $\tilde{\Phi}_p$
        $\boldsymbol{\mu}_p^* = \arg\min_{\boldsymbol{\mu}_p} \tilde{\Phi}_p(\boldsymbol{\mu}_p;\boldsymbol{\mu}^{(\tilde{n})})$
    end loop
    $\boldsymbol{\mu}^{(\tilde{n}+1)} = \begin{bmatrix}\boldsymbol{\mu}_1^* \\ \vdots \\ \boldsymbol{\mu}_P^*\end{bmatrix}$
end loop
```

Table 8.3: Skeleton of parallel algorithm for reducing inter-processor communication

8.3.1 Application to targeted reconstruction

An important application of the new cost function derived here is targeted reconstruction. In targeted reconstruction, the image volume $\boldsymbol{\mu}$ is divided into two disjoint regions, $\boldsymbol{\mu}_T$ and $\boldsymbol{\mu}_{T'}$ ($\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_T \\ \boldsymbol{\mu}_{T'}\end{bmatrix}$). $\boldsymbol{\mu}_T$ is the targeted image region, where the best-possible resolution afforded by the scanner is desired. The image volume outside the targeted volume is $\boldsymbol{\mu}_{T'}$, where the best possible reconstruction is not necessary to be achieved. When the above idea is applied to targeted reconstruction, we find that we can divide the objective function into two parts, $\tilde{\Phi}_T(\boldsymbol{\mu}_T;\boldsymbol{\mu}^{(\tilde{n})})$ and $\tilde{\Phi}_{T'}(\boldsymbol{\mu}_{T'};\boldsymbol{\mu}^{(\tilde{n})})$. We start two groups of threads.

In each thread group, the number of threads is equal to the number of processor cores. These thread groups are fairly scheduled by the operating system to run concurrently on the processor cores. We give $\tilde{\Phi}_T(\boldsymbol{\mu}_T; \boldsymbol{\mu}^{(\tilde{n})})$ to the first thread group, and $\tilde{\Phi}_{T'}(\boldsymbol{\mu}_{T'}; \boldsymbol{\mu}^{(\tilde{n})})$ to the second thread group. The second thread group quits early on most iterations, say every 4 out of 5, returning its initial image as its next iterate. So, in every 4 out of 5 iterations we devote all the processing power to the reconstruction of the targeted image region. The non-targeted image region is updated only once in every 5 iterations. Of course, experiments for a given application are required to test if this scheme is going to be advantageous. In any case, this setup is guaranteed to decrease the value of the overall cost function.

8.3.2 Overheads

While the proposed cost function reduces execution-time by reducing inter-processor communication, some overheads are also introduced. First overhead is the possibility of an increase in the number of iterations due to a possible reduction in convergence rate. The convergence rate could reduce because of two reasons. Firstly, some curvatures have to be increased to produce a valid quadratic surrogate. And secondly, due to a reduction in inter-processor communication, the algorithm updating a given image volume segment would be using older iterates of neighboring image volume segments. Another execution-time overhead is due to a forward-projection needed for those rays that cross image volume segment boundaries. This forward-projection is carried out once in the loop over p. It might be possible to avoid this overhead by using the fraction $\frac{[\boldsymbol{G}_p \boldsymbol{\mu}_{initial,p}]_i}{[\boldsymbol{G}\boldsymbol{\mu}_{initial}]_i}$, as an approximation to $\frac{[\boldsymbol{G}_p \boldsymbol{\mu}_p^{(\tilde{n})}]_i}{[\boldsymbol{G}\boldsymbol{\mu}^{(\tilde{n})}]_i}$. Memory overhead due to likelihood has two parts. The first part is due to the fact that every MPU must store all the rays passing through it. And the second part is due to the fact that every MPU must also store those pieces of the neighboring image volume segments that it needs for computing the parameters of $\tilde{\Phi}_p$. Memory overhead due to the penalty is insignificant when compared to the likelihood part : 2 image slices per neighbor per MPU.

CHAPTER 9

Conclusions and future work

9.1 Conclusions

There is a definite need for statistical image reconstruction methods in X-ray CT. However, these methods have to be made practical first. Various problems challenging the use of statistical methods in practical settings were studied in this thesis, and solutions to them proposed. Here are the conclusions that can be drawn from the proposed solutions.

- Imposition of the non-negativity constraint incurred a significant compute time penalty in PCG based methods. To avoid this compute time penalty, a modification of the Poisson negative log-likelihood was developed. The PCG algorithm minimizing the modified cost function could control negative pixels without a huge compute time overhead. Thus, we can conclude here that the non-negativity constraint does not impose an excessive compute-time overhead on PCG algorithms.
- A statistical image reconstruction method that produced reconstructions free of beam hardening artifacts and used the same beam hardening calibration information as traditional methods was developed. The practical use of previously known polyenergetic statistical methods was hindered due to their need for extra calibration information. The proposed method removed this hindrance. The conclusion here is that statistical methods can be made as practical as the traditional FBP methods with

respect to beam hardening correction.

- A brand new approach towards motion-compensated image reconstruction was proposed. It was shown to work for a limited, though challenging, class of simulated problems. No conclusions can be drawn regarding the utility of this method in practical situations, but the necessary first steps towards that goal were taken.
- Monotonicity is a desirable property that can be achieved in PCG based algorithms. But, monotonic PCG algorithms for non-quadratic cost functions were not known. And, non-quadratic cost functions are essential to prevent the edges from getting blurred in statistical reconstructions. A monotonic PCG algorithm for non-quadratic cost functions was tested in thesis. It provided monotonicity without sacrificing convergence rate. Thus, we can conclude here that PCG algorithms can actually provide a path to achieving the best possible reconstruction a cost function can offer.
- A comparison of convergence rates of OS and PCG based methods provided definite answers regarding the suitability of OS and PCG in different situations. The conclusion drawn from that comparison was that, starting from an FBP image, OS algorithms provide a faster convergence rate, but they become non-monotone after a few iterations. After the OS algorithm becomes non-monotone, monotone algorithms like PCG and ICD must be used to refine images further.
- A single algorithm that has all the desired properties does not exist till date. An attempt to create an algorithm with many desirable properties was made in the form of the hybrid OS-ICD algorithm. The hybrid algorithm approach shows promise, even though it has not met expectations yet.
- Parallel computation is a very direct approach to achieving a practical reconstruction time. Its usefulness to OS algorithms was proved beyond doubt.

- The parallel OS algorithm developed here had the shortcoming of frequent inter-processor communication. A new surrogate function that reduces the frequency of inter-processor communication while producing a monotonic reduction in the value of the original cost function was proposed. Thus, we can conclude that a monotone algorithm that can reduce inter-processor communication exists. Whether it will have a comparable convergence rate, remains to be shown.
- Two more algorithms were proposed to reduce the overall compute time. The first algorithm uses PCG techniques in an attempt to compute a search direction that is an improvement over the search direction computed by the OS algorithm. The second algorithm attempts to reduce the number of times a computationally expensive function is evaluated while still producing a monotonic decrease in the value of the cost function (up to the OS approximation) was proposed. These proposed algorithms indicate to us that solutions to current problems that are based on known techniques exist.

9.2 Future work

- The main theme of this thesis is practical application. Therefore, the evaluation of cost functions and image reconstruction algorithms developed here must be carried out for clinical datasets.
- Sophisticated regularization designs have been developed for bias-variance tradeoff in the past few years. Those designs should replace the space-invariant regularization and space-variant regularization of [26] used in this thesis.
- An important component of a statistical reconstruction method left unexplored in this thesis is the system model. While the distance-driven (DD) projector is fast, it makes certain approximations, which causes a small amount of artifact. A more

accurate system model with computational requirements comparable to DD needs to be developed.

- The use of the LBFGS-B algorithm to implement non-negativity constraint should be explored.
- Most algorithms explored here use the monoenergetic observation model, and they can be expanded easily to work with a polyenergetic model. The parameters used to approximate the nonlinearity due to beam hardening could also be jointly estimated from the observed data, instead of being measured beforehand.
- The motion-compensated statistical reconstruction method tested here is for simulated fan-beam scans. It must be adapted and tested for multislice CT scanners. Further, the use of this method for quasi-periodic motion like cardiac X-ray CT should be explored.
- Varous preconditioners are known in literature and their use in PCG based algorithms for X-ray CT should be explored. There might exist a high convergence-rate, monotonic algorithm based on PCG.
- An OS algorithm that behaves as a OS 256-subset algorithm in the middle slices of the reconstructed volume, and as a OS 41-subset algorithm in the end slices of helical cone-beam geometry could be found easily. This could make the hybrid OS-ICD algorithm a success.
- Further acceleration of the algorithms developed here can be done easily using more sophisticated features of the underlying hardware and software.
- Two surrogate functions that are proved to produce a monotonic decrease in the cost function value while reducing compute times in practical situations have been derived in Chapter 8. They must be implemented and their usefulness in practical situations

should be tested. A hybrid OS-PCG algorithm has also been derived in that chapter. A regularized algorithm based on it should be derived, implemented, and tested.

APPENDIX

APPENDIX A

Mathematical proofs

A.1 Proof for reduction of cost function value in the QS_PCG_LS algorithm

Lemma A.1.1. *For* $\phi^{(n,m)}(\boldsymbol{\mu})$ *and* $\Phi(\boldsymbol{\mu})$ *defined in Section 6.1.3 we have,*

$$\phi^{(n,m)}(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi^{(n,m)}(\boldsymbol{\mu}^{(n,m)}) \Rightarrow \Phi(\boldsymbol{\mu}^{(n+1,0)}) \leq \Phi(\boldsymbol{\mu}^{(n,0)})$$

Proof.

Define,

$$\begin{aligned}\phi_1(\boldsymbol{\mu}) &\stackrel{\triangle}{=} \phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}),\\ \phi_2(\boldsymbol{\mu}) &\stackrel{\triangle}{=} \phi^{(n,m)}(\boldsymbol{\mu}) = \phi_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,m)}).\end{aligned}$$

From the hypothesis, we have,

$$\phi_2(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi_2(\boldsymbol{\mu}^{(n,m)}),$$

$$\Rightarrow \phi_L(\boldsymbol{\mu}^{(n,m+1)}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}^{(n,m+1)}; \boldsymbol{\mu}^{(n,m)}) \leq \phi_L(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,m)}). \tag{A.1}$$

From the second part of the majorization conditions on ϕ_R (2.9),

$$R(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi_R(\boldsymbol{\mu}^{(n,m+1)}; \boldsymbol{\mu}^{(n,m)}).$$

Using this in the L.H.S. of (A.1),

$$\phi_L(\boldsymbol{\mu}^{(n,m+1)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi_L(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,0)}) + \phi_R(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,m)}).$$

From the first part of the majorization conditions on ϕ_R (2.8),

$$\phi_R(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,m)}) = R(\boldsymbol{\mu}^{(n,m)}).$$

Using this in the R.H.S. of the above inequality,

$$\phi_L(\boldsymbol{\mu}^{(n,m+1)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi_L(\boldsymbol{\mu}^{(n,m)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,m)}).$$

By applying the definition of ϕ_1, we have,

$$\phi_1(\boldsymbol{\mu}^{(n,m+1)}) \leq \phi_1(\boldsymbol{\mu}^{(n,m)}).$$

From the above inequality, we have, $\phi_1(\boldsymbol{\mu}^{(n,N_2)}) \leq \phi_1(\boldsymbol{\mu}^{(n,N_2-1)}) \leq \ldots \leq \phi_1(\boldsymbol{\mu}^{(n,0)})$.

From the definition of the algorithm , we have $\boldsymbol{\mu}^{(n,N_2)} = \boldsymbol{\mu}^{(n+1,0)}$.

We thus have,

$$\phi_1(\boldsymbol{\mu}^{(n+1,0)}) \leq \phi_1(\boldsymbol{\mu}^{(n,0)}).$$

Substituting the definition of ϕ_1 in the above inequality, we get,

$$\phi_L(\boldsymbol{\mu}^{(n+1,0)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n+1,0)}) \leq \phi_L(\boldsymbol{\mu}^{(n,0)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,0)}). \tag{A.2}$$

From the second part of the majorization conditions (2.9) on ϕ_L,

$$-L(\boldsymbol{\mu}^{(n+1,0)}) \leq \phi_L(\boldsymbol{\mu}^{(n+1,0)}; \boldsymbol{\mu}^{(n,0)}).$$

Using this in the L.H.S. of (A.2),

$$-L(\boldsymbol{\mu}^{(n+1,0)}) + R(\boldsymbol{\mu}^{(n+1,0)}) \leq \phi_L(\boldsymbol{\mu}^{(n,0)}; \boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,0)}).$$

From the first part of the majorization conditions on ϕ_L (2.8),

$$\phi_L(\boldsymbol{\mu}^{(n,0)};\boldsymbol{\mu}^{(n,0)}) = -L(\boldsymbol{\mu}^{(n,0)}).$$

Using this in the R.H.S. of the above inequality,

$$-L(\boldsymbol{\mu}^{(n+1,0)}) + R(\boldsymbol{\mu}^{(n+1,0)}) \leq -L(\boldsymbol{\mu}^{(n,0)}) + R(\boldsymbol{\mu}^{(n,0)}).$$

Using the definition of $\Phi(\boldsymbol{\mu})$, we get,

$$\Phi(\boldsymbol{\mu}^{(n+1,0)}) \leq \Phi(\boldsymbol{\mu}^{(n,0)}).$$

$\square$

A.2 Majorization proof for the quadratic surrogate of the modified cost function

The key result,*i.e.* the majorization proof, is in Lemma A.2.5. The proof is based on general facts about differentiable functions that are proved in Lemmas A.2.1,A.2.2, A.2.3 and A.2.4. The information about each of the terms of the negative log-likelihood that is used in Lemma A.2.5 is collected before hand in the preliminaries section below.

A.2.1 Preliminaries

The derivative of h_i can be found by differentiating (3.1) :

$$\dot{h}_i(t) = b_i e^{-t}\left(\frac{y_i}{b_i e^{-t} + r_i} - 1\right) \tag{A.3}$$

The derivative of $\tilde{h}_i$ can be found by differentiating (3.5) :

$$\dot{\tilde{h}}_i(t) = \begin{cases} \dot{h}_i(t) & \text{if } t \geq 0, \quad i \in I_1 \cup I_2 \cup I_3 \\ \dot{h}_i(0) & \text{if } t < 0, \quad i \in I_1 \cup I_2 \\ \dot{h}_i(0) + \frac{(y_i - r_i)^2}{y_i} t & \text{if } t < 0, \quad i \in I_3 \end{cases} \tag{A.4}$$

For simplicity, we drop the subscript i from h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ in the latter part of this sections. In the following sections we derive a few properties of h_i and obtain representative illustrations for h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for $i \in I_1, I_2$ and I_3.

$i \in I_1$ From (A.4), we have

$$\begin{aligned} \dot{\tilde{h}}(t) &= \dot{h}(t) && \text{for } t \geq 0 \\ &= \dot{h}(0) && \text{for } t < 0 \end{aligned}$$

Thus, for $t < 0$, $\dot{\tilde{h}}(t)$ is a constant with value $\dot{h}(0)$. From (A.3), we have $\dot{h}(0) = b_i(y_i/(b_i + r_i) - 1) \leq 0$. Since, $i \in I_1 \Rightarrow y_i \leq r_i \Rightarrow y_i \leq (b_i + r_i) \because b_i \geq 0$. For $t \geq 0$, $\dot{\tilde{h}}(t) = \dot{h}(t)$. From Lemma 1 of [1] we know that $\dot{h}(t)$ is strictly concave and monotonically increasing and

$$\begin{aligned} \lim_{t\to\infty} \dot{h}(t) &= \lim_{t\to\infty} b_i e^{-t}\left(\frac{y_i}{b_i e^{-t} + r_i} - 1\right) \\ &= 0. \end{aligned}$$

Thus, we get a representative plot of $\dot{\tilde{h}}_i$ in Fig. A.1. Similarly, representative plots of $\dot{h}_i, h_i$ and $\tilde{h}_i$ can be obtained.

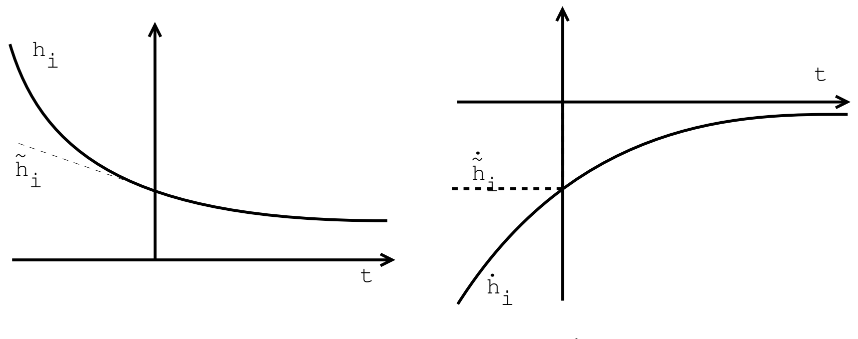


Figure A.1: Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for $i \in I_1$.

$i \in I_2$ From (A.4), we have

$$\begin{aligned} \dot{\tilde{h}}(t) &= \dot{h}(t) && \text{for } t \geq 0 \\ &= \dot{h}(0) && \text{for } t < 0 \end{aligned}$$

Property 1. Thus, for $t < 0$, $\dot{\tilde{h}}(t)$ is a constant with value $\dot{h}(0)$. From (A.3), we have $\dot{h}(0) = b_i(y_i/(b_i + r_i) - 1) \leq 0, \because i \in I_2 \Rightarrow y_i \leq (b_i + r_i)$.

Property 2. For $t \geq 0$, $\dot{\tilde{h}}(t) = \dot{h}(t)$. From Lemma 2 in [1], $\dot{h}(t)$ has exactly one maximizer, call it t^* ($= \ell^*$ in [1]).

$$\begin{aligned} t^* &= \log\left(\frac{b_i}{\sqrt{(y_i r_i)} - r_i}\right) \\ &= \log\left(\frac{b_i}{y_i - r_i}\right) + \log\left(1 + \frac{\sqrt{y}_i}{\sqrt{r}_i}\right) \\ &> 0 \qquad\qquad \because i \in I_2 \Rightarrow r_i < y_i \leq b_i + r_i \end{aligned}$$

Property 3. $\dot{h}(t^*) = (\sqrt{(y_i r_i)} - r_i)(\frac{\sqrt{y}_i}{\sqrt{r}_i} - 1) > 0, \because i \in I_2 \Rightarrow r_i < y_i \leq b_i + r_i$.

Property 4. From (P3) in Lemma 2 of [1], we have $\dot{h}(t)$ is strictly concave and monotonically increasing for $t < t^*$.

Property 5. From (P4) in Lemma 2 of [1], we have $\dot{h}(t)$ is monotonically decreasing for $t > t^*$.

Property 6.

$$\lim_{t\to\infty} \dot{h}(t) = \lim_{t\to\infty} b_i e^{-t}\left(\frac{y_i}{b_i e^{-t} + r_i} - 1\right) = 0$$

Combining properties 1…6, we get a representative plot of $\dot{\tilde{h}}_i$ in Fig. A.2. Similarly, representative plots of $\dot{h}_i$, h_i and $\tilde{h}_i$ can be obtained.

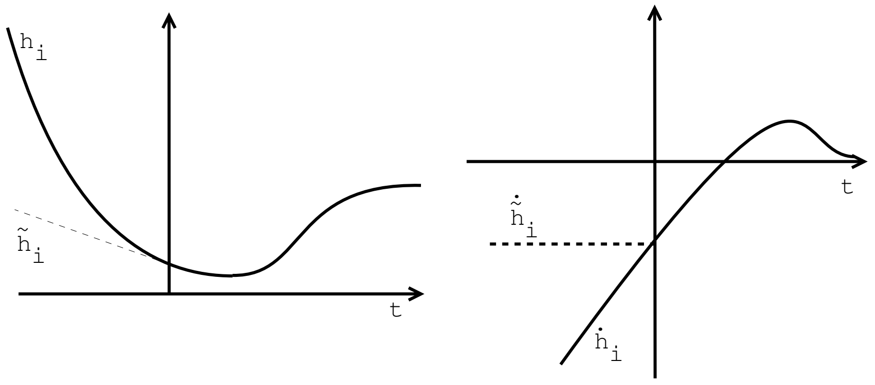


Figure A.2: Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_2.

$i \in I_3$ From (A.4), we have

$$\begin{aligned} \dot{\tilde{h}}(t) &= \dot{h}(t) && \text{for } t \geq 0 \\ &= \dot{h}_i(0) + \frac{(y_i - r_i)^2}{y_i} t && \text{for } t < 0 \end{aligned}$$

Property 7. Thus, for $t < 0$, $\dot{\tilde{h}}(t)$ is a straight line with a positive slope equal to $(y_i - r_i)^2/y_i$. From (A.3), we have $\dot{h}(0) = b_i(y_i/(b_i + r_i) - 1) > 0$, $\because i \in I_3 \Rightarrow y_i > (b_i + r_i)$. We can also safely assume $b_i \neq 0$; otherwise, $h(t)$ would be a constant.

Property 8. For $t \geq 0$, $\dot{\tilde{h}}(t) = \dot{h}(t)$. From Lemma 2 in [1], $\dot{h}(t)$ has exactly one maximizer, call it t^* ($= \ell^*$ in [1]).

$$\begin{aligned} t^* &= \log\left(\frac{b_i}{\sqrt{(y_i r_i)} - r_i}\right) \\ &= \log\left(\frac{b_i}{y_i - r_i}\right) + \log\left(1 + \frac{\sqrt{y}_i}{\sqrt{r}_i}\right) \\ &= t_1 + \log\left(1 + \frac{\sqrt{y}_i}{\sqrt{r}_i}\right) && \text{Let } t_1 = \log\left(\frac{b_i}{y_i - r_i}\right) \\ &\gtreqless 0 && \text{depending on } y_i, b_i \text{ and } r_i \end{aligned}$$

Property 9. $\dot{h}(t) = 0$ has a single real solution; call it t_1. Using (A.3), we get $t_1 = \log(b_i/(y_i - r_i)) < 0$, $\because i \in I_3 \Rightarrow y_i > b_i + r_i$.

Property 10. From property 8, we have $t^* > t_1, \because i \in I_3 \Rightarrow y_i > r_i$. Combining this with properties 8,4, 5 and 6 , we have $h(t) > 0, \forall t > 0 > t_1$.

Thus, when $t^* > 0$ we get Fig. A.3; when $t^* \leq 0$ we get Fig. A.4. Similarly, representative plots of $\dot{h}_i, h_i$ and $\tilde{h}_i$ can be obtained.

A.2.2 Results

Lemma A.2.1. *Given 1-D differentiable real functions $h(t)$ and $q(t; s)$ (where s is a fixed parameter), in order to satisfy $q(t; s) \geq h(t)$, $\forall t \geq s$ it is sufficient that $h(s) = q(s; s)$ and $\dot{q}(v; s) \geq \dot{h}(v)$, $\forall v \geq s$.*

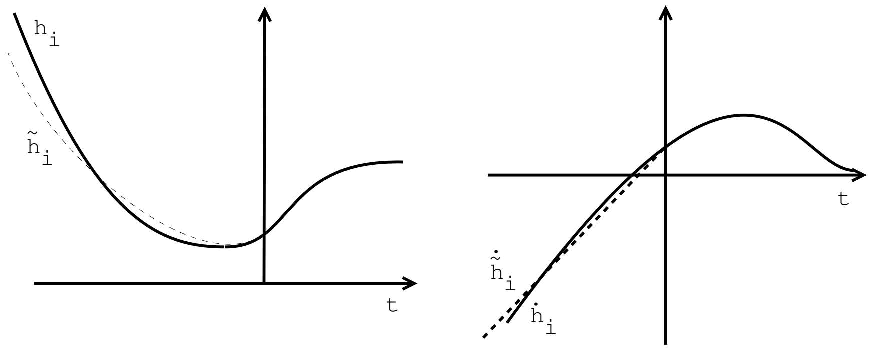


Figure A.3: Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_3 and maximum of $\dot{h}_i$ occurs in the positive quadrant.

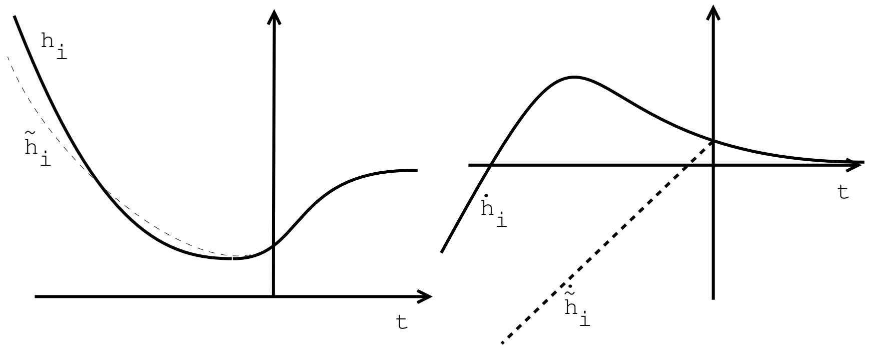


Figure A.4: Illustration of h_i, $\tilde{h}_i$, $\dot{h}_i$ and $\dot{\tilde{h}}_i$ for I_3 and maximum of $\dot{h}_i$ occurs in the negative quadrant.

Proof.

$$\begin{aligned}
h(t) &= h(s) + \int_s^t h(v)dv, \ \forall t \in \mathbb{R} \\
q(t;s) &= q(s;s) + \int_s^t q(v;s)dv, \ \forall t \in \mathbb{R} \\
\Rightarrow q(t;s) - h(t) &= \int_s^t \dot{q}(v;s) - \dot{h}(v)dv, \ (\because q(s;s) = h(s)) \\
&\geq 0, \ \forall t \geq s, (\because \dot{q}(v;s) \geq \dot{h}(v), \ \forall v \geq s).
\end{aligned}$$

□

Lemma A.2.2. *Given 1-D differentiable real functions $h(t)$ and $q(t;s)$ (where s is a fixed parameter), in order to satisfy $q(t;s) \geq h(t), \ \forall t \leq s$ it is sufficient that $h(s) = q(s;s)$ and $\dot{q}(v;s) \leq \dot{h}(v)$, $\forall v \leq s$.*

Proof. The proof is identical to Lemma A.2.1. □

Lemma A.2.3. *Given 1-D real functions $h(t)$ and $g(t)$, $h(t)$ is monotonically increasing over a set $S \subseteq \mathbb{R}$, $g(t)$ is monotonically decreasing over S and $\exists s \in S,$ such that $h(s) \geq g(s)$, then $h(t) \geq g(t),\ \forall t \geq s, t \in S$*

Proof.

It is given that, $h(s) \geq g(s)$. Since h is monotonically increasing,

$$h(t) \geq h(s),\ \forall t \geq s, t \in S$$

Since g is monotonically decreasing,

$$g(s) \geq g(t),\ \forall t \geq s, t \in S$$

$$\therefore h(t) \geq g(t),\ \forall t \geq s, t \in S$$

□

Lemma A.2.4. *Given 1-D real functions $h(t)$ and $g(t)$, $h(t)$ is monotonically increasing over a set $S \subseteq \mathbb{R}$, $g(t)$ is monotonically decreasing over S and $\exists s \in S, h(s) \leq g(s)$, then $h(t) \leq g(t),\ \forall t \leq s, t \in S$*

Proof. The proof is identical to Lemma A.2.3. □

Lemma A.2.5. *The function $\tilde{\phi}_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n,0)})$ (defined in (3.6)) majorizes $-\tilde{L}(\boldsymbol{\mu})$ (defined in (3.4))* i.e. *(2.8) and (2.9) hold.*
(For sake of simplicity, we drop 0 from $\boldsymbol{\mu}^{(n,0)}$.)

Proof.

<u>Part 1 : (2.8) holds</u>

$$\begin{aligned}
\boldsymbol{\mu} &= \boldsymbol{\mu}^{(n)} \\
\Rightarrow\ t &= s, && \forall i = 1, \cdots, n_d \text{ in (3.7)} \\
\Rightarrow\ q_{Li}(t; s) &= \tilde{h}_i(s) && \text{from (3.7)} \\
\therefore\ \tilde{\phi}_L(\boldsymbol{\mu}^{(n)}; \boldsymbol{\mu}^{(n)}) &= \sum_{i=1}^{n_d} q_{Li}([\boldsymbol{A\mu}^{(n)}]_i; [\boldsymbol{A\mu}^{(n)}]_i) \\
&= \sum_{i=1}^{n_d} \tilde{h}_i([\boldsymbol{A\mu}^{(n)}]_i) \\
&= \sum_{I_1 \cup I_2 \cup I_3} \tilde{h}_i([\boldsymbol{A\mu}^{(n)}]_i) \\
&= -\tilde{L}(\boldsymbol{\mu}^{(n)}) && \text{in (3.4)}
\end{aligned}$$

$\therefore$ (2.8) holds.

Part 2 : (2.9) holds

We need to prove $\tilde{\phi}_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) \geq -L(\boldsymbol{\mu}),\ \forall \boldsymbol{\mu} \in \mathbb{R}_p^n$. We rewrite $\tilde{\phi}_L$ in a form similar to (3.4) as follows :

$$\begin{aligned}
\tilde{\phi}_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) \quad = \quad & \sum_{i \in I_1} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) + \sum_{i \in I_2} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) \\
& + \sum_{i \in I_3} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)
\end{aligned} \tag{A.5}$$

and show that each of the terms in each of the above sums exceeds the corresponding term in (3.4).

We split the sums $\sum_{i \in I_3} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$ and $\sum_{i \in I_3} \tilde{h}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$ as follows :

$$\sum_{i \in I_3} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) = \sum_{i \in I'_3} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) + \sum_{i \in I''_3} q_{Li}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$$

$$\sum_{i \in I_3} \tilde{h}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) = \sum_{i \in I'_3} \tilde{h}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i) + \sum_{i \in I''_3} \tilde{h}([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$$

$$\text{where} I'_3 = \{i \in I_3 |\ t_i^* \geq 0\}$$

$$I''_3 = \{i \in I_3 |\ t_i^* < 0\}$$

Though the form of the surrogate is the same for both I'_3 and I''_3 the proofs are slightly different. We handle I'_3 case in case 3 and I''_3 in case 4. For simplicity, we drop the subscript i from h_i, $\tilde{h}_i$, $\dot{h}_i$, $\dot{\tilde{h}}_i$, $\ddot{h}_i$ and $\breve{c}_i$; similarly, we drop Li from q_{Li}.

Case 1 *($i \in I_1$).*

From (A.4)

$$\dot{\tilde{h}}(t) = \dot{h}(t) \qquad \text{if } t \geq 0$$
$$= \dot{h}(0) \qquad \text{if } t < 0$$

Case 1.1 *($s \leq 0$).*

From (3.7),

$$q(t; s) = \tilde{h}(t) + \dot{\tilde{h}}(s)(t - s) + \tfrac{1}{2}\ddot{h}(0)(t - s)^2$$

$$\Rightarrow \dot{q}(t; s) = \dot{\tilde{h}}(s) + \ddot{h}(0)(t - s)$$

$$\text{Note} : \dot{q}(s; s) = \dot{\tilde{h}}(s)$$

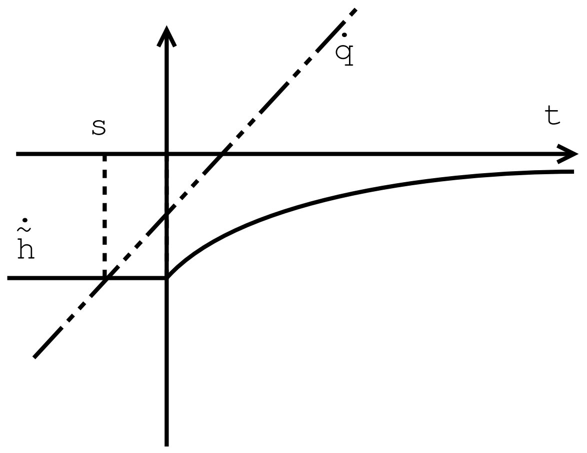


Figure A.5: Representative illustration for Case 1.1($i \in I_1, s \leq 0$).

Case 1.1.1 *($s \leq 0, t < s$).*

$$\begin{aligned} &\text{Now, } \dot{\tilde{h}}(t) = \dot{h}(0) = \dot{\tilde{h}}(s) \\ &\quad \dot{q}(t;s) = \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s) \\ &\Rightarrow \dot{q}(t;s) < \dot{\tilde{h}}(t) \qquad\qquad \forall t < s \end{aligned}$$

By Lemma A.2.2, $q(t;s) \geq h(t),\ \forall t < s$.

Case 1.1.2 *($s \leq 0, s \leq t \leq 0$).*

$$\begin{aligned} &\text{Now, } \dot{\tilde{h}}(t) = \dot{h}(0) = \dot{\tilde{h}}(s) \\ &\quad \dot{q}(t;s) = \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s) \\ &\Rightarrow \dot{q}(t;s) \geq \dot{\tilde{h}}(t) \qquad\qquad \forall s \leq t \leq 0 \end{aligned}$$

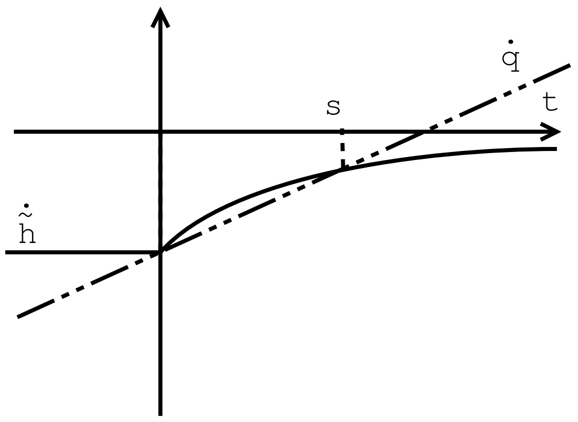


Figure A.6: Representative illustration for Case 1.2($i \in I_1, s > 0$).

Case 1.1.3 *($s \leq 0, 0 < t$).*

$$\begin{aligned}
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s) \\
&= \dot{q}(0;s) + \ddot{h}(0)t \\
&= \dot{q}(0;s) + \int_0^t \ddot{h}(0)dv \\
&> \dot{q}(0;s) + \int_0^t \ddot{h}(v)dv && \because t > 0 \\
&\because \ddot{h}(v) < \ddot{h}(0), \text{for } v \geq 0 \text{ by (E3) of Lemma 2 of [1]} \\
&> \dot{\tilde{h}}(0) + \int_0^t \ddot{h}(v)dv && \text{from Case 1.1.2} \\
&= \dot{\tilde{h}}(0) + \int_0^t \ddot{\tilde{h}}(v)dv && \text{by definition of } \tilde{h} \text{ (t)} \\
&= \dot{\tilde{h}}(t) && \forall t > 0
\end{aligned}$$

By combining cases 1.1.2 and 1.1.3 we satisfy conditions for Lemma A.2.1. $\therefore q(t;s) \geq h(t), \forall t \geq s$.

By combining the above cases, we observe that $q(t;s) \geq h(t), \forall s \leq 0$.

Case 1.2 *($s > 0$).*

From (3.7),

$$q(t;s) = \dot{h}(s) + \dot{\tilde{h}}(s)(t-s) + \tfrac{1}{2}\frac{\dot{h}(s) - \dot{h}(0)}{s}(t-s)^2$$
$$\Rightarrow \dot{q}(t;s) = \dot{\tilde{h}}(s) + \frac{\dot{h}(s) - \dot{h}(0)}{s}(t-s)$$

Also, $\frac{\dot{h}(s)-\dot{h}(0)}{s} > 0$(using (E4) of Lemma 2 of [1]).

Case 1.2.1 *($s > 0, t \geq s$).*

$$\dot{q}(0;s) = \dot{h}(0)$$
$$\dot{q}(s;s) = \dot{\tilde{h}}(s) = \dot{h}(s)$$

From the formulation of $\dot{q}(t;s)$, we know that it is a straight line. From above, we know that it intersects $\dot{h}(t)$ at 0 and s. $\dot{h}(t)$ is strictly concave by (E3) of Lemma 2 of [1]. Using Lemma 4 of [1], we have,

$$\dot{q}(t;s) \geq \dot{h}(t) = \dot{\tilde{h}}(t),\ \forall t \geq s$$

Using lemma A.2.1, we have

$$q(t;s) \geq \tilde{h}(t),\ \forall t \geq s$$

Case 1.2.2 *($s > 0, 0 \leq t < s$).*

Using arguments from Case 1.2.1, we have

$$\dot{q}(t;s) \leq \dot{h}(t) = \dot{\tilde{h}}(t),\ \forall 0 \leq t < s$$

Case 1.2.3 *($s > 0, t < 0$).*

$$\begin{aligned}
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \frac{\dot{h}(s) - \dot{h}(0)}{s}(t-s) \\
&= \dot{q}(0;s) + \frac{\dot{h}(s) - \dot{h}(0)}{s}t \\
&= \dot{\tilde{h}}(0) + \frac{\dot{h}(s) - \dot{h}(0)}{s}t && \text{using Case 1.2.1 and } \dot{\tilde{h}}(0) = \dot{h}(0) \\
&< \dot{\tilde{h}}(0) \\
&= \dot{\tilde{h}}(t) && \text{by definition of } \dot{\tilde{h}}(t) \\
\therefore \dot{q}(t;s) &\le \dot{\tilde{h}}(t), \forall t < 0
\end{aligned}$$

From cases 1.2.2 and 1.2.3, we have $\dot{q}(t;s) \le \dot{\tilde{h}}(t), \forall t < s$. Using Lemma A.2.2, we have $q(t;s) \ge \dot{\tilde{h}}(t), \forall t < s$. Using the above result and combining with case 1.2.1, we have,

$$q(t;s) \ge \dot{\tilde{h}}(t), \forall s > 0$$

Using cases 1.1 and 1.2, we have,

$$q(t,s) \ge \dot{\tilde{h}}(t), \forall t, s \in \mathbb{R}$$

From (A.4),

Case 2 *($i \in I_2$).*

$$\dot{\tilde{h}}(t) = \begin{cases} \dot{h}(t) & \text{if } t \ge 0 \\ \dot{h}(0) & \text{if } t < 0 \end{cases}$$

Case 2.1 *($s \le 0$).*

Fig. A.7 shows a representative illustration for case 2.1. From (3.7),

$$\begin{aligned}
q(t;s) &= \tilde{h}(s) + \dot{\tilde{h}}(s)(t-s) + \tfrac{1}{2}\ddot{h}(0)(t-s)^2 \\
\Rightarrow \dot{q}(t;s) &= \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s)
\end{aligned}$$

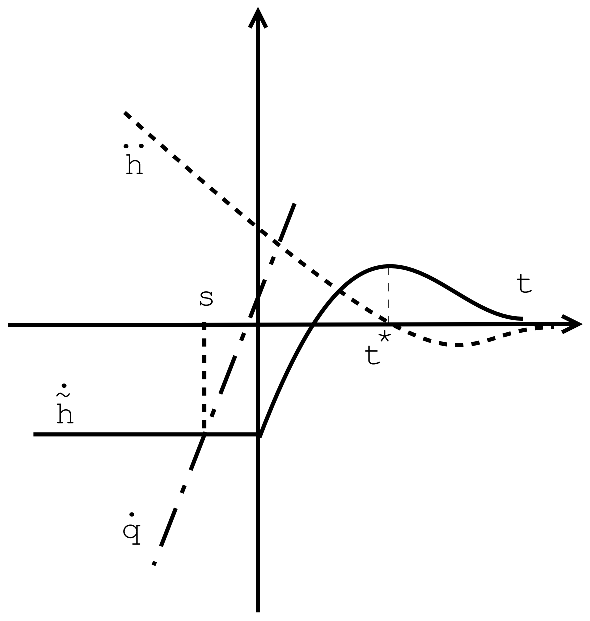


Figure A.7: Representative illustration for Case 2.1($i \in I_2, s \leq 0$).

Case 2.1.1 *($s \leq 0, t < s$).*

$$\dot{q}(t;s) = \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s)$$

$$= \dot{h}(0) + \ddot{h}(0)(t-s) \qquad \text{by definition of } \dot{\tilde{h}} \text{ (s)}$$

Using equation (30) of [1],

$$\ddot{h}(0) = b_i(1 - \frac{y_i r_i}{(b_i + r_i)^2}) \geq 0$$

$$\Rightarrow \dot{q}(t;s) \leq \dot{h}(0)$$

$$= \dot{\tilde{h}}(t) \qquad \text{by definition of } \dot{\tilde{h}} \text{ (t)}$$

From Lemma A.2.2, we have

$$q(t;s) \geq \tilde{h}(t), \forall t < s$$

Case 2.1.2 *($s \leq 0, s \leq t \leq 0$).*

Using arguments of case 2.1.1,

$$\begin{aligned}\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \ddot{h}(0)(t-s) \\ &\geq \dot{\tilde{h}}(t)\end{aligned}$$

Case 2.1.3 *($s \leq 0, 0 < t \leq t^*$).*

See Section A.2.1 for the definition of t^*. Using facts from Lemma 2 of [1], we have $\ddot{h}(t)$ is monotonically decreasing for $t < t^z$ (= ℓ^z in [1]). Also, $t^* < t^z$. $\therefore 0 \leq t \leq t^* \Rightarrow \ddot{h}(0) \geq \ddot{h}(t) \geq \ddot{h}(t^*)$.

$$\begin{aligned}\dot{q}(t;s) &= \dot{q}(0;s) + \int_0^t \ddot{q}(v;s)dv \\ &= \dot{q}(0;s) + \int_0^t \ddot{h}(0)dv \\ &\geq \dot{q}(0;s) + \int_0^t \ddot{h}(v)dv && \text{see above} \\ &= \dot{\tilde{h}}(s) + \ddot{h}(0)(-s) + \int_0^t \ddot{h}(v)dv && \text{substituting for } \dot{q}(0;s) \\ &\geq \dot{\tilde{h}}(s) + \int_0^t \ddot{h}(v)dv && \because \ddot{h}(0) \geq 0, s \leq 0 \\ &= \dot{\tilde{h}}(0) + \int_0^t \ddot{h}(v)dv && \text{by definition of } \tilde{h}(s) \\ &= \dot{\tilde{h}}(0) + \int_0^t \ddot{\tilde{h}}(v)dv && \text{by definition of } \tilde{h}(s) \\ &= \dot{\tilde{h}}(t)\end{aligned}$$

Case 2.1.4 *($s \leq 0, t^* < t$).*

From case 2.1.3, we have $\dot{q}(t;s) \geq \dot{\tilde{h}}(t^*)$. $\dot{q}(t;s)$ is monotonically increasing because $\ddot{h}(0) > 0$. $\dot{\tilde{h}}(t) = \dot{h}(t)$ is monotonically decreasing for $t > t^*$. By Lemma A.2.3, we have

$$\dot{q}(t;s) \geq \dot{\tilde{h}}(t),\ \forall t > t^*$$

Combining cases 2.1.2 …2.1.4, we have $\dot{q}(t;s) \geq \dot{h}(t)$, $\forall t \geq s$. By Lemma A.2.1,

$$q(t;s) \geq h(t), \forall t \geq s$$

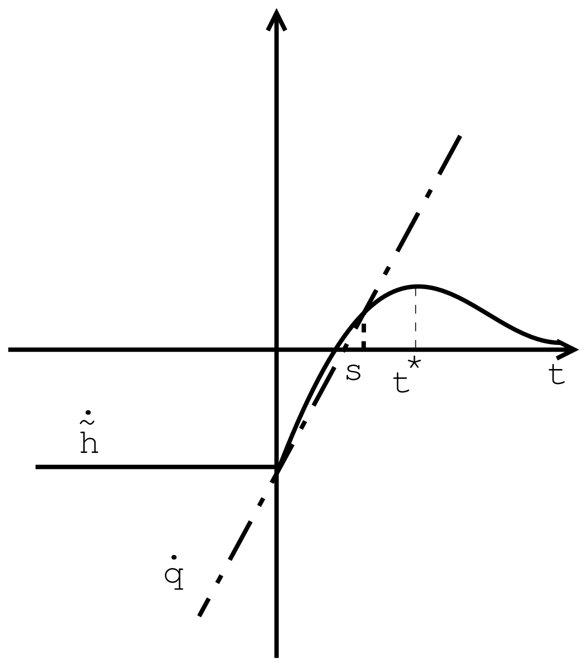


Figure A.8: Representative illustration for Case 2.2($i \in I_2, 0 < s \leq t^*$).

Combining this with case 2.1.1,

$$q(t; s) \geq h(t), \forall t \in \mathbb{R}, s \leq 0$$

Case 2.2 *(*$0 < s \leq t^*$*).*

$$\begin{aligned} q(t; s) &= \dot{h}(s) + \dot{\tilde{h}}(s)(t - s) + \tfrac{1}{2}\frac{\dot{h}(s) - \dot{h}(0)}{s}(t - s)^2 \\ \dot{q}(t; s) &= \dot{\tilde{h}}(s) + \frac{\dot{h}(s) - \dot{h}(0)}{s}(t - s) \end{aligned}$$

Case 2.2.1 *(*$0 < s \leq t^*, 0 \leq t \leq s$*).*

Here, $\dot{\tilde{h}}(t) = \dot{h}(t)$. From (P3) of lemma 2 of [1],$\dot{h}$ (t) is concave over $0 \leq t \leq t^*$. Now, $\dot{q}(0; s) = \dot{\tilde{h}}(s) - \frac{\dot{h}(s) - \dot{h}(0)}{s} s = \dot{h}(0) = \dot{\tilde{h}}(0)$. Also, $\dot{q}(s; s) = \dot{\tilde{h}}(s)$. Thus, by formulation $\dot{q}(t; s)$ is a straight line that intersects a concave curve $\dot{\tilde{h}}(t)$ at 0 and s. Thus, by Lemma 4 of [1] we have,

$$\dot{q}(t; s) \leq \dot{\tilde{h}}(t), \forall 0 \leq t \leq s$$

Case 2.2.2 *(*$0 < s \leq t^*, s < t \leq t^*$*).*

Arguments in case 2.2.1 are applicable here. Thus, by Lemma 4 of [1] we have,

$$\dot{q}(t;s) \geq \dot{\tilde{h}}(t), \forall s \leq t < t^*$$

Now, $\dot{q}(t;s)$ and $\dot{\tilde{h}}(t)$ are continuous at t^* and $\dot{q}(t;s) \geq \dot{\tilde{h}}(t)$, $\forall s \leq t < t^*$. Therefore,

$$\lim_{t \to t^*} \dot{q}(t;s) \geq \lim_{t \to t^*} \dot{\tilde{h}}(t).$$

Therefore, $\dot{q}(t^*;s) \geq \dot{\tilde{h}}(t^*)$.

Case 2.2.3 *(*$0 < s \leq t^*, t^* < t$*).*

From (P3) of Lemma 2 of [1], we have $\dot{h}(s) > \dot{h}(0)$. Therefore, $(\dot{h}(s) - \dot{h}(0))/s > 0$. Thus, $\dot{q}(t;s)$ is a monotonically increasing function .From (P4) of Lemma 2 of [1], we have $\dot{\tilde{h}}(t)$ is a monotonically decreasing function. Using lemma A.2.3,

$$\dot{q}(t;s) \geq \dot{\tilde{h}}(t), \forall t > t^*$$

Case 2.2.4 *(*$0 < s \leq t^*, t < 0$*).*

$$\begin{aligned}
\dot{q}(t;s) &= \dot{q}(0;s) + \int_0^t \ddot{q}(v;s)dv \\
&= \dot{h}(0) + \int_0^t \ddot{q}(v;s)dv && \text{from case 2.2.1} \\
&= \dot{\tilde{h}}(t) + \int_0^t \ddot{q}(v;s)dv && \text{definition of } \dot{\tilde{h}} \text{ (t)} \\
&= \dot{\tilde{h}}(t) + \frac{\dot{h}(s) - \dot{h}(0)}{s} t \\
&< \dot{\tilde{h}}(t) && \because \frac{\dot{h}(s) - \dot{h}(0)}{s} > 0, t < 0
\end{aligned}$$

Combining cases 2.2.1 and 2.2.4, $\dot{q}(t;s) \leq \dot{\tilde{h}}(t),\ \forall t \leq s$. Using lemma A.2.2 we have, $q(t;s) \geq \dot{\tilde{h}}(t),\ \forall t \leq s$. Combining cases 2.2.2 and 2.2.3, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t),\ \forall t > s$. Using lemma A.2.1 we have, $q(t;s) \geq \dot{\tilde{h}}(t),\ \forall t > s$. Thus,

$$q(t;s) \geq \dot{\tilde{h}}(t), \forall t \in \mathbb{R}, 0 < s < t^*$$

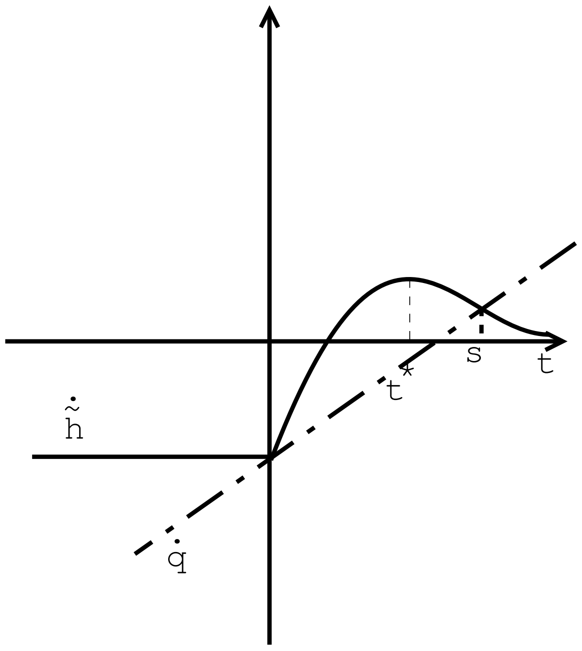


Figure A.9: Representative illustration for Case 2.3($i \in I_2, t^* < s$).

Case 2.3 *($t^* < s$).*

$$q(t;s) = \dot{h}(s) + \dot{\tilde{h}}(s)(t-s) + \tfrac{1}{2}\frac{\dot{h}(s) - \dot{h}(0)}{s}(t-s)^2$$
$$\dot{q}(t;s) = \dot{\tilde{h}}(s) + \frac{\dot{h}(s) - \dot{h}(0)}{s}(t-s)$$

From (A.3), $\dot{h}(0) = b_i(y_i/(b_i + r_i) - 1) \leq 0$, since $i \in I_2 \Rightarrow y_i \leq (b_i + r_i)$. From property 3 we have, $\dot{h}(t^*) > 0$. From (P4) of Lemma 2 of [1], we have $\dot{h}(t)$ is monotonically decreasing for $t > t^*$. From property 6 we have, $\lim_{t\to\infty} \dot{h}(t) = 0$. Therefore, $\dot{h}(t) > 0, \forall t > t^*$, since $t_1 < t^*$ from property 8. Therefore, $\dot{h}(s) - \dot{h}(0) > 0$ and so $(\dot{h}(s) - \dot{h}(0))/s > 0$.

Case 2.3.1 *($t^* < s, s \leq t$).*

By definition of $\dot{q}(t;s)$, $\dot{q}(s;s) = \dot{\tilde{h}}(s)$. $\dot{q}(t;s)$ is monotonically increasing because $(\dot{h}(s) - \dot{h}(0))/s > 0$. From (P4) of Lemma 2 of [1], we have $\dot{\tilde{h}}(t)$ is monotonically decreasing. By lemma A.2.3, $\dot{q}(t;s) \geq \dot{\tilde{h}}(s), \forall t \geq s$. Using lemma A.2.1 we have,

$$q(t;s) \geq h(t), \forall t \geq s$$

Case 2.3.2 *($t^* < s, t^* \leq t < s$).*

By definition of $\dot{q}(t;s)$, $\dot{q}(s;s) = \dot{\tilde{h}}(s)$. $\dot{q}(t;s)$ is monotonically increasing because $(\dot{h}(s) - \dot{h}(0))/s > 0$. From (P4) of Lemma 2 of [1], we have $\dot{\tilde{h}}(t)$ is monotonically decreasing. By lemma A.2.4, $\dot{q}(t;s) \geq \dot{\tilde{h}}(s), \forall t \geq s$. Using lemma A.2.2 we have,

$$q(t;s) \geq h(t), \forall t^* \leq t < s$$

Case 2.3.3 *($t^* < s, 0 \leq t < t^*$).*

We define, $r(t) = \dot{h}(0) + \frac{\dot{h}(t^*) - \dot{h}(0)}{t^*}t$. Note that, $\dot{q}(0;s) = r(0)$. $\dot{h}(t^*) - \dot{h}(0) > \dot{h}(s) - \dot{h}(0)$, since $\dot{h}(s)$ is monotonically decreasing for $s > t^*$ (by (P4) of Lemma 2 of [1]). Therefore, $(\dot{h}(t^*) - \dot{h}(0))/t^* > (\dot{h}(s) - \dot{h}(0))/s$. Thus,

$$r(t) \geq \dot{q}(t),\ \forall t \geq 0.$$

Now, $\dot{\tilde{h}}(0) = r(0)$ and $\dot{\tilde{h}}(t^*) = r(t^*)$. Also, $\dot{\tilde{h}}(t)$ is strictly concave for $0 < t < t^*$ and intersects the line $r(t)$ at 0 and t^*. By Lemma 4 of [1] we have,

$$\dot{\tilde{h}}(t) \geq r(t),\ 0 \leq t \leq t^*.$$

Therefore,

$$\dot{\tilde{h}}(t) \geq \dot{q}(t;s),\ 0 \leq t < t^*$$

Case 2.3.4 *($t^* < s, t < 0$).*

By definition of $\dot{q}(t;s)$,

$$\begin{aligned}
\dot{q}(t;s) &= \dot{h}(0) + \frac{\dot{h}(s) - \dot{h}(0)}{s}t \\
&< \dot{h}(0) && \because \frac{\dot{h}(s) - \dot{h}(0)}{s} > 0, t < 0 \\
&< \dot{\tilde{h}}(t) && \text{by definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

By combining cases 2.3.2 … 2.3.4 we have $\dot{q}(t;s) \leq \dot{\tilde{h}}(t), \forall t < s$. By lemma A.2.2 we have,

$$q(t;s) \geq h(t), \forall t < s.$$

Combining this with case 2.3.1 we have

$$q(t;s) \geq \tilde{h}(t), \forall t \in \mathbb{R}, t^* < s$$

Combining cases 2.1, 2.2 and 2.3 we have

$$q(t;s) \geq \dot{\tilde{h}}(t),\ \forall t, s \in \mathbb{R} \text{ and } i \in I_2$$

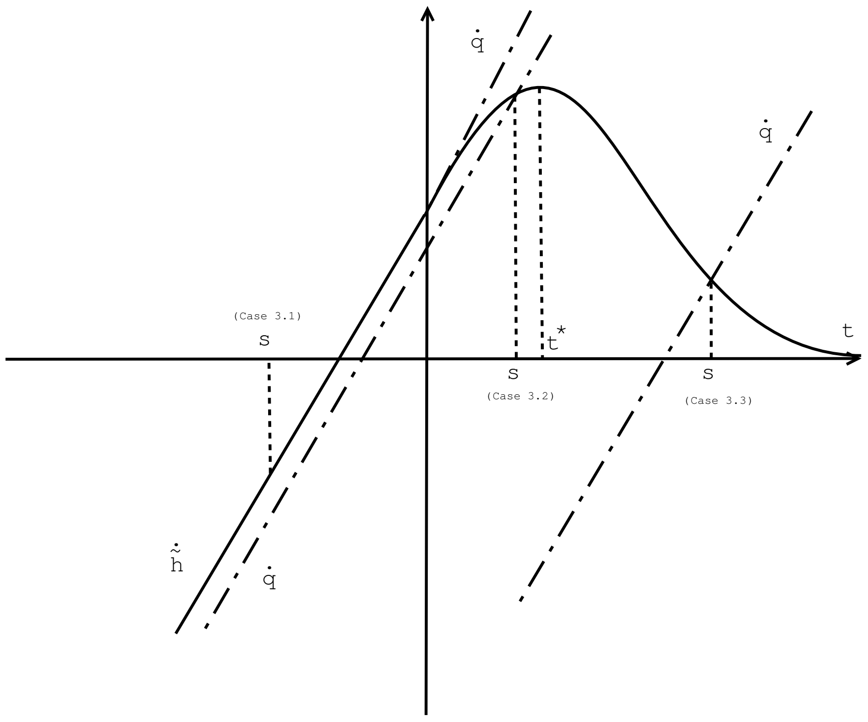


Figure A.10: Representative illustrations for Case 3.1($i \in I'_3, s \leq 0$), Case 3.2($i \in I'_3, 0 < s \leq t^*$) and Case 3.3($i \in I'_3, t^* < s$).

Case 3 *(*$i \in I'_3$*).*

$$\tilde{h}(t) = \begin{cases} h(t) & \text{if } t \geq 0 \\ h(0) + \dot{h}(0)t + \frac{1}{2}\frac{(y_i - r_i)^2}{y_i}t^2 & \text{if } t < 0 \end{cases}$$

$$\dot{\tilde{h}}(t) = \begin{cases} \dot{h}(t) & \text{if } t \geq 0 \\ \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}t & \text{if } t < 0 \end{cases}$$

$$q(t;s) = h(s) + \dot{\tilde{h}}(s)(t-s) + \frac{1}{2}\frac{(y_i - r_i)^2}{y_i}(t-s)^2$$

$$\dot{q}(t;s) = \dot{\tilde{h}}(s) + \frac{(y_i - r_i)^2}{y_i}(t-s)$$

Case 3.1 *($s \leq 0$).*

Case 3.1.1 *($s \leq 0, t \leq 0$).* By definition of $q(t; s)$,

$$\begin{aligned}
\dot{q}(t; s) &= \dot{\tilde{h}}(s) + \frac{(y_i - r_i)^2}{y_i}(t - s) \\
&= \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}(s) + \frac{(y_i - r_i)^2}{y_i}(t - s) && \text{substituting for } \dot{\tilde{h}}(s) \\
&= \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}t \\
&= \dot{\tilde{h}}(t) && \text{from definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

Applying lemma A.2.2 for $t \leq s$, we have

$$q(t; s) \geq h(t), t \leq s, s \leq 0$$

Case 3.1.2 *($s \leq 0, 0 < t \leq t^*$).*

$$\begin{aligned}
\dot{q}(t; s) &= \dot{q}(0; s) + \int_0^t \ddot{q}(v; s)dv \\
&= \dot{\tilde{h}}(s) - \frac{(y_i - r_i)^2}{y_i}s + \int_0^t \ddot{q}(v; s)dv && \text{substituting for } \dot{q}(0; s) \\
&= \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}s - \frac{(y_i - r_i)^2}{y_i}s + \int_0^t \ddot{q}(v; s)dv && \text{substituting for } \dot{\tilde{h}}(s) \\
&= \dot{\tilde{h}}(0) + \int_0^t \ddot{q}(v; s)dv && \text{from definition of } \dot{\tilde{h}}
\end{aligned}$$

From property 9 we have,

$$\begin{aligned}
t_1 &= \log(b_i/(y_i - r_i)) \\
\Rightarrow e^{-t_1} &= \frac{y_i - r_i}{b_i} \\
\ddot{h}(t) &= (1 - \frac{y_i r_i}{(b_i e^{-t} + r_i)^2})b_i e^{-t} && \text{From equation (30) of [1]} \\
\Rightarrow \ddot{h}(t_1) &= \frac{(y_i - r_i)^2}{y_i}
\end{aligned}$$

From property 9 we know that $t_1 < 0$. From lemma 2 of [1] we know that $\ddot{h}(t)$ is monotonically decreasing for $t \leq t^*$. Thus, $t_1 < 0 \leq v \leq t^* \Rightarrow \ddot{h}(t_1) > \ddot{h}(0) \geq \ddot{h}(v) \geq \ddot{h}(t^*)$.

$$\begin{aligned}
\therefore \dot{q}(t;s) &\geq \dot{\tilde{h}}(0) + \int_0^t \ddot{h}(v)dv \\
&\geq \dot{\tilde{h}}(0) + \int_0^t \ddot{\tilde{h}}(v)dv && \text{from definition of } \tilde{h}(t) \\
&\geq \dot{\tilde{h}}(t)
\end{aligned}$$

Case 3.1.3 *(*$s \leq 0, t^* < t$*).*

From case 3.1.2 we have, $\dot{q}(t^*;s) \geq \dot{\tilde{h}}(t^*)$. $\dot{q}(t;s)$ is monotonically increasing since it is a straight line and its slope $\frac{(y_i - r_i)^2}{y_i} > 0$. From (P4) of lemma 2 of [1]we know that, $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t > t^*$. Using lemma A.2.3 we have,

$$\dot{q}(t;s) \geq \dot{\tilde{h}}(t), t > t^*$$

Combining cases 3.1.1 …3.1.3 we have, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t;s), t \geq s$. Using lemma A.2.1, $q(t;s) \geq \tilde{h}(t;s), t \geq s$. Combining this with the result in case 3.1.1 we have,

$$q(t;s) \geq \tilde{h}(t;s), \forall t \in \mathbb{R}, s \leq 0$$

Case 3.2 *(*$0 < s \leq t^*$*).*

Case 3.2.1 *(*$0 < s \leq t^*, s \leq t \leq t^*$*).*

$$\begin{aligned}
\dot{q}(t;s) &= \dot{q}(s;s) + \int_s^t \ddot{q}(v;s)dv \\
&= \dot{\tilde{h}}(s) + \int_s^t \ddot{h}(v;s)dv && \text{from case 3.1.2, } t_1 < 0 \leq v \leq t^* \Rightarrow \\
& && \ddot{h}(t_1) > \ddot{h}(0) \geq \ddot{h}(v) \geq \ddot{h}(t^*) \\
&\geq \dot{\tilde{h}}(s) + \int_s^t \ddot{\tilde{h}}(v;s)dv && \text{from definition of } \tilde{h}(t) \\
&\geq \dot{\tilde{h}}(t)
\end{aligned}$$

Thus, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t), s \leq t \leq t^*, 0 < s \leq t^*$.

Case 3.2.2 *($0 < s \leq t^*, t^* < t$).*

From case 3.2.1, $\dot{q}(t^*; s) \geq \dot{\tilde{h}}(t^*)$. From (P4) of lemma 2 of [1], $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t > t^*$. By lemma A.2.3, $\dot{q}(t; s) \geq \dot{\tilde{h}}(t), t^*m < t$.

Combining cases 3.2.1 and 3.2.2 we have, $\dot{q}(t; s) \geq \dot{\tilde{h}}(t), t \geq s$. By lemma A.2.1,

$$q(t; s) \geq \dot{\tilde{h}}(t), t \geq s, 0 < s \leq t^*$$

Case 3.2.3 *($0 < s \leq t^*, 0 \leq t < s$).*

$$\begin{aligned}
\dot{q}(t; s) &= \dot{\tilde{h}}(s) + \ddot{h}(t_1)(t - s) && \text{using expression for } \ddot{h}(t_1) \text{ from case 3.1.2}\\
&= \dot{\tilde{h}}(s) - \int_t^s \ddot{h}(t_1) dv\\
&\leq \dot{\tilde{h}}(s) - \int_t^s \ddot{h}(v; s) dv && \ddot{h}(s) \text{ is monotonically decreasing}\\
& && \text{for } t < t^*, \text{ ,from case 3.1.2}\\
&= \dot{\tilde{h}}(s) + \int_s^t \ddot{h}(v; s) dv && \ddot{h}(s)\\
&= \dot{\tilde{h}}(t)
\end{aligned}$$

Case 3.2.4 *($0 < s \leq t^*, t < 0$).* From case 3.2.3 we know that, $\dot{q}(0; s) \leq \dot{\tilde{h}}(0)$.

$$\begin{aligned}
\dot{q}(t; s) &= \dot{\tilde{h}}(s) + \ddot{h}(t_1)(t - s)\\
&= \dot{q}(0; s) + \ddot{h}(t_1)t && \text{substituting for } \dot{q}(0; s)\\
&\leq \dot{\tilde{h}}(0) + \ddot{h}(t_1)t && \text{see above}\\
&= \dot{\tilde{h}}(t) && \text{from definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

Combining cases 3.2.3 and 3.2.4 we have, $\dot{q}(t; s) \leq \dot{\tilde{h}}(t), \forall t \leq s$. By lemma 2 of [1], $q(t; s) \geq \dot{\tilde{h}}(t), t \leq s$

Combining the results from case 3.2.2 and 3.2.4,

$$q(t; s) \geq \tilde{h}(t), \forall t \in \mathbb{R}, 0 < s < t^*$$

Case 3.3 *($t^* < s$).*

Case 3.3.1 *($t^* < s, s \leq t$).*

$\dot{q}(t;s)$ is monotonically increasing since $\frac{(y_i-r_i)^2}{y_i} > 0$. By (P4) of lemma 2 of [1], $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t \geq s > t^*$. By their respective definitions, $\dot{q}(s;s) = \dot{\tilde{h}}(s)$. By lemma A.2.3 we have, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t), \forall t \geq s$. By lemma A.2.1 we have, $q(t;s) \geq \tilde{h}(t), \forall t \geq s$.

Case 3.3.2 *($t^* < s, t^* \leq t < s$).*

$\dot{q}(t;s)$ is monotonically increasing since $\frac{(y_i-r_i)^2}{y_i} > 0$. By (P4) of lemma 2 of [1], $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t^* \leq t < s$. By their respective definitions, $\dot{q}(s;s) = \dot{\tilde{h}}(s)$. By lemma A.2.4 we have, $\dot{q}(t;s) \leq \dot{\tilde{h}}(t), \forall t^* \leq t < s$.

Case 3.3.3 *($t^* < s, 0 \leq t < t^*$).*

$$\begin{aligned}
\dot{q}(t;s) &= \dot{q}(t^*;s) + \int_{t^*}^{t} \ddot{q}(v;s)dv \\
&\leq \dot{\tilde{h}}(t^*) - \int_{t}^{t^*} \ddot{q}(v;s)dv && \text{from case 3.2.3} \\
&= \dot{\tilde{h}}(t^*) - \int_{t}^{t^*} \ddot{h}(t_1)dv && \text{using expression for } \ddot{h}(t_1) \text{ from case 3.1.2} \\
&\leq \dot{\tilde{h}}(t^*) - \int_{t}^{t^*} \ddot{h}(v;s)dv && \ddot{h}(s) \text{ is monotonically decreasing} \\
& && \text{for } t < t^*, \text{ ,from case 3.1.2} \\
&= \dot{\tilde{h}}(s) + \int_{t}^{t^*} \ddot{h}(v;s)dv \\
&= \dot{\tilde{h}}(t)
\end{aligned}$$

Case 3.3.4 *($t^* < s, t < 0$).* From case 3.3.3 we have, $\dot{q}(0;s) \leq \dot{\tilde{h}}(0)$.

$$\begin{aligned}
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \ddot{h}(t_1)(t-s) \\
&= \dot{q}(0;s) + \ddot{h}(t_1)t && \text{substituting for } \dot{q}(0;s) \\
&\leq \dot{\tilde{h}}(0) + \ddot{h}(t_1)t && \text{see above} \\
&= \dot{\tilde{h}}(t) && \text{from definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

Combining cases 3.3.2, 3.3.3 and 3.3.4 we have, $\dot{q}(t;s) \leq \dot{\tilde{h}}(t), \forall t \leq s$. By lemma A.2.2, $q(t;s) \geq \dot{\tilde{h}}(t), \forall t \leq s$.

Combining results from case 3.3.1 and 3.3.4 we have,

$$q(t;s) \geq \tilde{h}(t), \forall t \in \mathbb{R}, t^* < s$$

Combining results from cases 3.1 ... 3.3 we have,

$$q(t;s) \geq \tilde{h}(t), i \in {I'}_3$$

Case 4 *($i \in {I''}_3$).*

$$\begin{aligned}
\tilde{h}(t) &= \begin{cases} h(t) & \text{if } t \geq 0 \\ h(0) + \dot{h}(0)t + \frac{1}{2}\frac{(y_i-r_i)^2}{y_i}t^2 & \text{if } t < 0 \end{cases} \\
\dot{\tilde{h}}(t) &= \begin{cases} \dot{h}(t) & \text{if } t \geq 0 \\ \dot{h}(0) + \frac{(y_i-r_i)^2}{y_i}t & \text{if } t < 0 \end{cases} \\
q(t;s) &= h(s) + \dot{\tilde{h}}(s)(t-s) + \tfrac{1}{2}\frac{(y_i-r_i)^2}{y_i}(t-s)^2 \\
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \frac{(y_i-r_i)^2}{y_i}(t-s)
\end{aligned}$$

Case 4.1 *($s \leq 0$).*

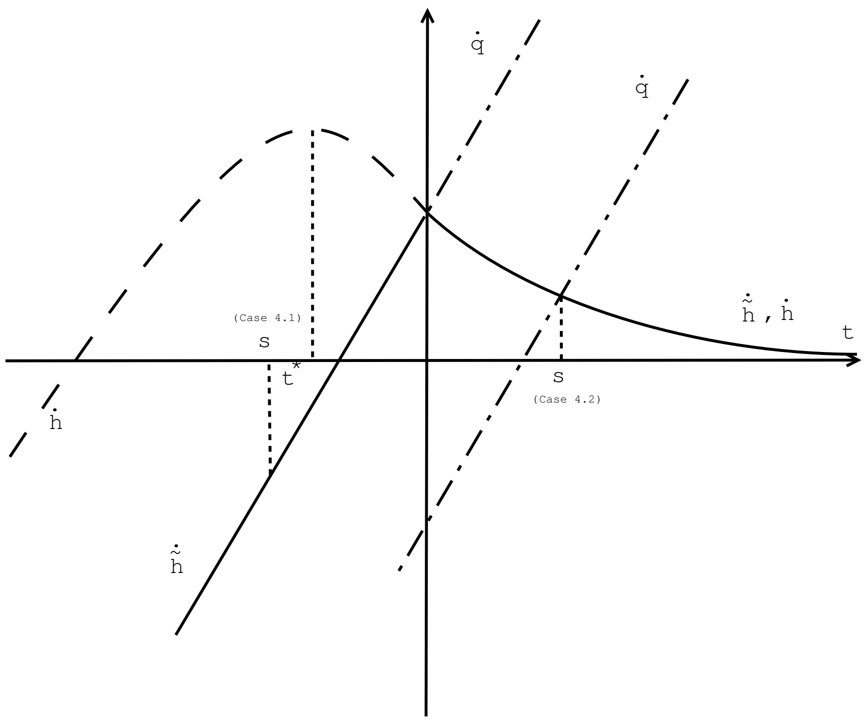


Figure A.11: Representative illustrations for Case 4.1($i \in I''_3, s \leq 0$) and Case 4.2($i \in I''_3, 0 < s$).

Case 4.1.1 *($s \leq 0, t \leq 0$).* By definition of $q(t; s)$,

$$\begin{aligned}
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \frac{(y_i - r_i)^2}{y_i}(t-s) \\
&= \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}s + \frac{(y_i - r_i)^2}{y_i}(t-s) && \text{substituting for } \dot{\tilde{h}}(s) \\
&= \dot{h}(0) + \frac{(y_i - r_i)^2}{y_i}t \\
&= \dot{\tilde{h}}(t) && \text{from definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

Applying lemma A.2.2 for $t \leq s$, we have

$$q(t; s) \geq h(t), t \leq s, s \leq 0$$

Case 4.1.2 *($s \leq 0, 0 < t$).*

From case 4.1.1 we have, $\dot{q}(0; s) \geq \dot{\tilde{h}}(0)$. $\dot{q}(t; s)$ is monotonically increasing since it is a straight line and its slope $\frac{(y_i - r_i)^2}{y_i} > 0$. From (P4) of lemma 2 of [1]we know that, $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t > 0 > t^*$. Using lemma A.2.3 we have,

$$\dot{q}(t;s) \geq \dot{\tilde{h}}(t), t > 0$$

Combining cases 4.1.1 …4.1.2 we have, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t;s), t \geq s$. Using lemma A.2.1, $q(t;s) \geq \tilde{h}(t;s), t \geq s$. Combining this with the result in case 4.1.1 we have,

$$q(t;s) \geq \tilde{h}(t;s), \forall t \in \mathbb{R}, s \leq 0$$

Case 4.2 *($0 < s$).*

Case 4.2.1 *($0 < s, s \leq t$).*
$\dot{q}(t;s)$ is monotonically increasing since $\frac{(y_i-r_i)^2}{y_i} > 0$. By (P4) of lemma 2 of [1], $\dot{\tilde{h}}(t)$ is monotonically decreasing for $t \geq s > 0 > t^*$. By lemma A.2.3 we have, $\dot{q}(t;s) \geq \dot{\tilde{h}}(t), \forall t \geq s$. By their respective definitions, $\dot{q}(s;s) = \dot{\tilde{h}}(s)$. By lemma A.2.1 we have, $q(t;s) \geq \tilde{h}(t), \forall t \geq s$.

Case 4.2.2 *($0 < s, 0 \leq t < s$).*
$\dot{q}(t;s)$ is monotonically increasing since $\frac{(y_i-r_i)^2}{y_i} > 0$. By (P4) of lemma 2 of [1], $\dot{\tilde{h}}(t)$ is monotonically decreasing for $0 \leq t < s$. By lemma A.2.4 we have, $\dot{q}(t;s) \leq \dot{\tilde{h}}(t), \forall 0 \leq t < s$.

Case 4.2.3 *($0 < s, t < 0$).* From case 4.2.2 we have, $\dot{q}(0;s) \leq \dot{\tilde{h}}(0)$.

$$\begin{aligned}
\dot{q}(t;s) &= \dot{\tilde{h}}(s) + \ddot{h}(t_1)(t-s) \\
&= \dot{q}(0;s) + \ddot{h}(t_1)t && \text{substituting for } \dot{q}(0;s) \\
&\leq \dot{\tilde{h}}(0) + \ddot{h}(t_1)t && \text{see above} \\
&= \dot{\tilde{h}}(t) && \text{from definition of } \dot{\tilde{h}}(t)
\end{aligned}$$

Combining cases 4.2.2 and 4.2.3 we have, $\dot{q}(t;s) \leq \dot{\tilde{h}}(t), \forall t \leq s$. By lemma A.2.2, $q(t;s) \geq \dot{\tilde{h}}(t), \forall t \leq s$.

Combining results from case 4.2.1 and 4.2.3 we have,

$$q(t; s) \geq \tilde{h}(t), \forall t \in \mathbb{R}, 0 < s$$

Combining results from cases 4.1 and 4.2 we have,

$$q(t; s) \geq \tilde{h}(t), i \in I''_3$$

From cases 1, 2, 3 and 4 we see that each term of $\tilde{\phi}_L([\boldsymbol{A\mu}]_i; [\boldsymbol{A\mu}^{(n)}]_i)$ exceeds each term of $-\tilde{L}(\boldsymbol{\mu})$ for all values of $\boldsymbol{\mu} \in S = \mathbb{R}_p^n$. Thus,

$$\tilde{\phi}_L(\boldsymbol{\mu}; \boldsymbol{\mu}^{(n)}) \geq -\tilde{L}(\boldsymbol{\mu}), \forall \boldsymbol{\mu} \in S = \mathbb{R}_p^n$$

$\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] H. Erdoğan and J. A. Fessler. Monotonic algorithms for transmission tomography. *IEEE Trans. Med. Imag.*, 18(9):801–14, September 1999.

[2] H. Erdoğan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Phys. Med. Biol.*, 44(11):2835–51, November 1999.

[3] S. Ahn, J. A. Fessler, D. Blatt, and A. O. Hero. Convergent incremental optimization transfer algorithms: Application to tomography. *IEEE Trans. Med. Imag.*, 25(3):283–96, March 2006.

[4] A. Averbuch, D. L. Donoho, R. R. Coifman, M. Israeli, and Y. Shkolnisky. Fast slant stack: A notion of Radon transform for data in cartesian grid which is rapidly computable, algebrically exact, geometrically faithful and invertible. 2001. From http://www-stat.stanford.edu/~donoho/reports.html.

[5] J. M. Bardsley and C. R. Vogel. A nonnegatively constrained convex programming method for image reconstruction. *SIAM J. Sci. Comp.*, 25(4):1326–43, 2003.

[6] S. Basu and Y. Bresler. $O(n^2 \log_2(n))$ filtered backprojection reconstruction algorithm for tomography. *IEEE Trans. Im. Proc.*, 9(10):1760–73, October 2000.

[7] S. Basu and Y. Bresler. Error analysis and performance optimization of fast hierarchical backprojection algorithms. *IEEE Trans. Im. Proc.*, 10(7):1103–17, July 2001.

[8] F. J. Beekman and C. Kamphuis. Ordered subset reconstruction for x-ray CT. *Phys. Med. Biol.*, 46(7):1835–55, July 2001.

[9] D. Blatt, A. O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM J. Optim.*, 18(1):29–51, 2 2007.

[10] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. Technical Report NAM-08, Dept. EECS, Northwestern Univ., May 1994.

[11] R. H. Chan and W-K. Ching. Toeplitz-circulant preconditioners for Toeplitz systems and their applications to queueing networks with batch arrivals. *SIAM J. Sci. Comp.*, 17(3):762–72, May 1996.

[12] C. R. Crawford, K. F. King, C. J. Ritchie, and J. D. Godwin. Respiratory compensation in projection imaging using a magnification and displacement model. *IEEE Trans. Med. Imag.*, 15(3):327–32, June 1996.

[13] B. De Man and S. Basu. Distance-driven projection and backprojection in three dimensions. *Phys. Med. Biol.*, 49(11):2463–75, June 2004.

[14] B. De Man, J. Nuyts, P. Dupont, G. Marchal, and P. Suetens. An iterative maximum-likelihood polychromatic algorithm for CT. *IEEE Trans. Med. Imag.*, 20(10):999–1008, October 2001.

[15] A. R. De Pierro. A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography. *IEEE Trans. Med. Imag.*, 14(1):132–137, March 1995.

[16] M. Defrise, F. Noo, and H. Kudo. A solution to the long-object problem in helical cone-beam tomography. *Phys. Med. Biol.*, 45(3):623–43, March 2000.

[17] A. C. Dhanantwari, S. Stergiopoulos, and I. Iakovidis. Correcting organ motion artifacts in x-ray CT medical imaging systems by adaptive processing. I. Theory. *Med. Phys.*, 28(8):1562–76, August 2001.

[18] A. C. Dhanantwari, S. Stergiopoulos, N. Zamboglou, D. Baltas, H-G. Vogt, and G. Karangelis. Correcting organ motion artifacts in x-ray CT systems based on tracking of motion phase by the spatial overlap correlator. II. Experimental study. *Med. Phys.*, 28(8):1577–96, August 2001.

[19] I. A. Elbakri. *Statistical reconstruction algorithms for polyenergetic X-ray computed tomography.* PhD thesis, Univ. of Michigan, Ann Arbor, MI, 48109-2122, Ann Arbor, MI, 2003.

[20] I. A. Elbakri and J. A. Fessler. Statistical image reconstruction for polyenergetic X-ray computed tomography. *IEEE Trans. Med. Imag.*, 21(2):89–99, February 2002.

[21] I. A. Elbakri and J. A. Fessler. Segmentation-free statistical image reconstruction for polyenergetic X-ray computed tomography with experimental validation. *Phys. Med. Biol.*, 48(15):2543–78, August 2003.

[22] L. A. Feldkamp, L. C. Davis, and J. W. Kress. Practical cone beam algorithm. *J. Opt. Soc. Am. A*, 1(6):612–9, June 1984.

[23] J. A. Fessler. Hybrid Poisson/polynomial objective functions for tomographic image reconstruction from transmission scans. *IEEE Trans. Im. Proc.*, 4(10):1439–50, October 1995.

[24] J. A. Fessler. *Image reconstruction: Algorithms and analysis.* 2006. Book in preparation.

[25] J. A. Fessler and S. D. Booth. Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction. *IEEE Trans. Im. Proc.*, 8(5):688–99, May 1999.

[26] J. A. Fessler and W. L. Rogers. Spatial resolution properties of penalized-likelihood image reconstruction methods: Space-invariant tomographs. *IEEE Trans. Im. Proc.*, 5(9):1346–58, September 1996.

[27] P. Grangeat. Mathematical framework of cone beam 3D reconstruction via the first derivative of the Radon transform. In 1990 Oberwolfach, editor, *Mathematical methods in tomography*, pages 66–95. Springer, Berlin, 1991.

[28] M. R. Hestenes. *The conjugate direction methods in optimization.* Springer-Verlag, New York, 1981.

[29] S. Horbelt, M. Liebling, and M. Unser. Discretization of the Radon transform and of its inverse by spline convolutions. *IEEE Trans. Med. Imag.*, 21(4):363–76, April 2002.

[30] J. Hsieh. Adaptive streak artifact reduction in computed tomography resulting from excessive x-ray photon noise. *Med. Phys.*, 25(11):2139–47, November 1998.

[31] J. Hsieh. *Computed tomography: Principles, design, artifacts, and recent advances.* SPIE, Bellingham, 2003.

[32] J. Hsieh, O. E. Gurmen, and K. F. King. Investigation of a solid-state detector for advanced computed tomography. *IEEE Trans. Med. Imag.*, 19(9):930–40, September 2000.

[33] J. Hsieh, J. Li, J. Londt, K. Mohr, M. Vass, X. Tang, and D. Okerlund. Step-and-shoot VCT cardiac imaging. In *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, volume 4, pages 2368–71, 2005.

[34] P. J. Huber. *Robust statistics.* Wiley, New York, 1981.

[35] H. M. Hudson and R. S. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Trans. Med. Imag.*, 13(4):601–9, December 1994.

[36] D. R. Hunter and K. Lange. A tutorial on MM algorithms. *American Statistician*, 58(1):30–7, February 2004.

[37] C. A. Johnson, J. Seidel, and A. Sofer. Interior point methodology for 3-D PET reconstruction. *IEEE Trans. Med. Imag.*, 19(4):271–85, April 2000.

[38] P. M. Joseph and R. D. Spital. A method for correcting bone induced artifacts in computed tomography scanners. *J. Comp. Assisted Tomo.*, 2(1):100–8, January 1978.

[39] M. Kachelriess, D.-A. Sennst, W. Maxlmoser, and W. A. Kalender. Kymogram detection and kymogram-correlated image reconstruction from subsecond spiral computed tomography scans of the heart. *Med. Phys.*, 29(7):1489–1503, 2002.

[40] A. Katsevich. Theoretically exact filtered backprojection-type inversion algorithm for spiral CT. *SIAM J. Appl. Math.*, 62(6):2012–26, 2002.

[41] D. B. Keesing and J. A. O. Sullivan. Parallelization of a fully 3D CT iterative reconstruction algorithm. In *Proc. IEEE Intl. Symp. Biomed. Imag.*, pages 1240–3, 2006.

[42] J. S. Kole and F. J. Beekman. Parallel statistical image reconstruction for cone-beam x-ray CT on a shared memory computation platform. *Phys. Med. Biol.*, 50(6):1265–72, March 2005.

[43] P. J. La Riviere, J. Bian, and P. A. Vargas. Penalized-likelihood sinogram restoration for computed tomography. *IEEE Trans. Med. Imag.*, 25(8):1022–36, August 2006.

[44] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *J. Computational and Graphical Stat.*, 9(1):1–20, March 2000.

[45] D. G. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, Massachusetts, 2 edition, 1984.

[46] A. Macovski. *Medical imaging systems*. Prentice-Hall, New Jersey, 1983.

[47] R. Manzke. *Cardiac cone beam CT*. PhD thesis, Division of imaging sciences, King's college London, United Kingdom, September 2004.

[48] F. Natterer. Sampling in fan beam tomography. *SIAM J. Appl. Math.*, 53(2):358–80, April 1993.

[49] T. Nielsen, R. Manzke, R. Proksa, and M. Grass. Cardiac cone-beam ct volume reconstruction using art. *Med. Phys.*, 32(4):851–60, April 2005.

[50] R. Noumeir, G. E. Mailloux, and R. Lemieux. Detection of motion during tomographic acquisition by an optical flow algorithm. *Comput. Biomed. Res.*, 29(1):1–15, February 1996.

[51] R. Pan and S. J. Reeves. Fast Huber-Markov edge-preserving image restoration. In *Proc. SPIE 5674, Computational Imaging III*, pages 138–46, 2005.

[52] R. Proksa, T. Kohler, M. Grass, and J. Timmer. The n-pi-method for helical cone-beam ct. *Medical Imaging, IEEE Transactions on*, 19(9):848–63, September 2000.

[53] A. G. Ramm and A. I. Katsevich. *The Radon transform and local tomography*. CRC Press, Boca Raton, 1996.

[54] V. Raptopoulos, A. Karellas, J. Bernstein J, F. R. Reale, C. Constantinou, and J. K. Zawacki. Value of dual-energy ct in differentiating focal fatty infiltration of the liver from low-density masses. *AJR Am J Roentgenol.*, 157(4):721–5, 10 1991.

[55] S. J. Reeves. Fast restoration of PMMW imagery without boundary artifacts. In *Proc. SPIE 4719, Infrared and passive millimeter-wave imaging systems: design, analysis, modeling, and testing*, pages 289–95, 2002.

[56] K. Sauer and C. Bouman. A local update strategy for iterative reconstruction from projections. *IEEE Trans. Sig. Proc.*, 41(2):534–48, February 1993.

[57] H. Shi and J. A. Fessler. Quadratic regularization design for fan beam transmission tomography. In *Proc. SPIE 5747, Medical Imaging 2005: Image Proc.*, pages 2023–33, 2005.

[58] B. D. Smith. Image reconstruction from cone-beam projections - necessary and sufficient conditions and reconstruction methods. *IEEE Trans. Med. Imag.*, 4(1):14–25, March 1985.

[59] S. Srivastava and J. A. Fessler. Simplified statistical image reconstruction algorithm for polyenergetic X-ray CT. In *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, volume 3, pages 1551–5, 2005.

[60] K. C. Tam, S. Samarasekera, and F. Sauer. Exact cone beam ct with a spiral scan. *Physics in Medicine and Biology*, 43(4):1015–24, April 1998.

[61] J-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh. A three-dimensional statistical approach to improved image quality for multi-slice helical CT. *Med. Phys.*, 34(11):4526–44, November 2007.

[62] T. L. Toth, E. Cesmeli, A. Ikhlef, and T. Horiuchi. Image quality and dose optimization using novel X-ray source filters tailored to patient size. In *spie-5745*, pages 283–91, 2005.

[63] H. Turbell and P-E. Danielsson. Helical cone-beam tomography. *Intl. J. Imaging Sys. and Tech.*, 11(1):91–100, 2000.

[64] H. K. Tuy. An inversion formula for cone-beam reconstruction. *SIAM J. Appl. Math.*, 43(3):546–52, June 1983.

[65] H. K. Tuy. 3d image reconstruction for helical partial cone beam scanners. In *1999 international meeting on fully three-dimensional image reconstruction in radiology and nuclear medicine*, 1999.

[66] C. R. Vogel and M. E. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Trans. Im. Proc.*, 7(6):813–24, June 1998.

[67] D. J. Walter, E. J. Tkaczyk, and X. Wu. Accuracy and precision of dual energy CT imaging for the quantification of tissue fat content. In *Proc. SPIE 6142, Medical Imaging 2006: Phys. Med. Im.*, page 61421G, 2006.

[68] A. Yendiki and J. A. Fessler. A comparison of rotation- and blob-based system models for 3D SPECT with depth-dependent detector response. *Phys. Med. Biol.*, 49(11):2157–68, June 2004.

[69] W. Zbijewski and F. J. Beekman. Efficient monte carlo based scatter artifact reduction in cone-beam micro-ct. *IEEE Trans. Med. Imag.*, 25(7):817–27, July 2006.

[70] Y. Zhang-O'Connor and J. A. Fessler. Fourier-based forward and back-projectors in iterative fan-beam tomographic image reconstruction. *IEEE Trans. Med. Imag.*, 25(5):582–9, May 2006.