ACCELERATED OPTIMIZATION ALGORITHMS FOR STATISTICAL 3D X-RAY COMPUTED TOMOGRAPHY IMAGE RECONSTRUCTION

by

Donghwan Kim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2014

Doctoral Committee:

Professor Jeffrey A. Fessler, Chair
Assistant Professor Laura K. Balzano
Associate Professor Marina A. Epelman
Professor Alfred O. Hero III

ACKNOWLEDGEMENTS

My last five years in Ann Arbor pursuing this dissertation have been fulfilling and unforgettable and this would not have been possible without people around me who deserve thanks. Most importantly, I would like to thank my advisor, Prof. Jeffrey A. Fessler, a hundred times. I really enjoyed working with him as his insight and advice provided me wonderful research experience as well as great outcomes. He has encouraged every time when I felt stupid when facing research problems and that really helped me make myself confident in my work and ability.

Next, I would like to acknowledge my doctoral committee for their time and suggestions on improving this dissertation. The research collaboration with GE Healthcare and GE Global Research Center was effective and I would like to thank them for all their support including the real CT data used in this dissertation.

Inside the research group, I was glad to have a seat right next to Dr. Sathish Ramani in the office, who shared great insights on my research. I also would like to thank all of the group members for their inspiring research and kindness, and particularly I want to point out Hung Nien and Madison McGaffin for helpful discussion on our common research topic.

Outside of my research, I will always be grateful for endless support from my family. I should list each of their names here: my mother Okkyung Eun, my father Janghee Kim, and my sister Dongyoung Kim. I would also like to thank my friends in Korea who constantly kept in touch with me. In Ann Arbor, friends from my first year and last year, Korean EECS friends, and a soccer team AK United were always around me and made my life in Ann Arbor joyful.

Finally, I should not forget to acknowledge invaluable financial support from the Department of EECS, KLA-Tencor Corporation and National Institutes of Health, and equipment support from Intel Corporation.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

LIST OF ABBREVIATIONS

ABCD Axial block coordinate descent

BCD Block coordinate descent

BI Bounded interval

CD Coordinate descent

CG Conjugate gradient

CT Computed tomography

DD Distance-driven

DRA Dynamic range adjustment

EM Expectation maximization

FBP Filtered back-projection

FDK Feldkamp-Davis-Kress reconstruction

FGM Fast gradient methods

FO First-order methods

FWHM Full-width half-maximum

GCD Grouped coordinate descent

GD Gradient descent

GE General Electric

GEPP GE performance phantom

GM Gradient method

HBM Heavy-ball method

ICD Iterative coordinate descent

IOT Incremental optimization transfer

NH Non-homogeneous

NRMSD Normalized root-mean squared difference

NU Non-uniform

OGM Optimized gradient methods

OS Ordered subsets

OT Optimization transfer

PCG Preconditioned conjugated gradient

PEP Performance estimation problem

PET Positron emission tomography

PL Penalized likelihood

PWLS Penalized weighted least squares

QS Quadratic surrogates

QSQS Quasi-separable quadratic surrogates

RAM Random access memory

RMSD Root-mean squared difference

ROI Region-of-interest

SDP Semidefinite programming

SF Separable footprint

SPECT Single-photon emission computed tomography

SQS Separable quadratic surrogates

SS Separable surrogates

VS Variable-splitting

ABSTRACT

ACCELERATED OPTIMIZATION ALGORITHMS FOR STATISTICAL 3D X-RAY COMPUTED TOMOGRAPHY IMAGE RECONSTRUCTION

by

Donghwan Kim

Chair: Jeffrey A. Fessler

X-ray computed tomography (CT) has been widely celebrated for its ability to visualize the anatomical information of patients, but has been criticized for high radiation exposure. Statistical image reconstruction algorithms in X-ray CT can provide improved image quality for reduced dose levels in contrast to the conventional reconstruction methods like filtered back-projection (FBP). However, the statistical approach requires substantial computation time, more than half an hour for commercial 3D X-ray CT products. Therefore, this dissertation focuses on developing iterative algorithms for statistical reconstruction that converge within fewer iterations and that are amenable to massive parallelization in modern multi-processor implementations.

Ordered subsets (OS) methods have been used widely in tomography problems, because they reduce the computational cost by using only a subset of the measurement data per iteration. This dissertation first improves OS methods so that they better handle 3D helical cone-beam CT geometries. OS methods have been used in commercial positron emission tomography (PET) and single-photon emission CT (SPECT) since 1997. However, they require too long a reconstruction time in X-ray CT to be used routinely for every clinical CT scan. In this dissertation, two main approaches are proposed for accelerating OS algorithms, one that uses new optimization transfer approaches and one that combines OS with momentum algorithms.

First, the separable quadratic surrogates (SQS) methods, a widely used optimization transfer method with OS methods yielding simple, efficient and massively parallelizable OS-SQS methods, have been accelerated in three different ways; a nonuniform SQS (NU-SQS),

a SQS with bounded interval (SQS-BI), and a quasi-separable quadratic surrogates (QSQS) method. Among them, a new NU-SQS method that encourages larger step sizes for the voxels that are expected to change more between the current and the final image has highly accelerated the convergence, while the derivation guarantees monotonic descent.

Next, we combined OS methods with momentum approaches that cleverly reuse previous updates with almost negligible increased computation. Using momentum approaches such as well-known Nesterov's methods were found to be not fast enough for 3D X-ray CT reconstruction, but the proposed combination of OS methods and momentum approaches (OS-momentum) resulted in very fast convergence rates. OS-momentum algorithms sometimes suffered from instability, we adapted relaxed momentum schemes. This refinement improves stability without losing the fast rate of OS-momentum. To further accelerate OS-momentum algorithms, this dissertation proposes new momentum methods, called optimized gradient methods, which are twice as fast yet have remarkably simple implementations comparable to Nesterov's methods.

Finally, in addition to OS-type algorithms, one variant of the block coordinate descent (BCD) algorithm, called axial BCD (ABCD), is specifically designed for 3D cone-beam CT geometry. The chosen axial block of voxels in 3D CT geometry for the BCD algorithm enables both accelerated convergence and efficient computation, particularly when designed hand-in-hand with separable footprint (SF) projector.

Overall, this dissertation proposes several promising accelerated iterative algorithms for 3D X-ray CT image reconstruction. Their performance are investigated on simulated and real patient 3D CT scans.

CHAPTER I

Introduction

X-ray computed tomography (CT) has been widely celebrated for its ability to visualize the anatomical information of patients, but has been criticized for high radiation exposure [15]. Statistical image reconstruction methods can improve resolution and reduce noise and artifacts even for reduced dose levels [92], unlike conventional filtered back-projection (FBP) methods, by minimizing either penalized likelihood (PL) [1, 32, 36] or penalized weighted least-squares (PWLS) [96,102,103] cost functions that model the physics and statistics in X-ray CT. The primary drawback of these methods is their computationally expensive iterative algorithms. Therefore, we propose new accelerated optimization algorithms for 3D X-ray CT statistical image reconstruction.

Over the last few decades, many iterative algorithms have been adapted and developed for accelerated X-ray statistical image reconstruction. Coordinate descent (CD) [14, 109], preconditioned conjugate gradient (PCG) [38, 45], expectation-maximization (EM) [25, 28] and ordered subsets (OS) [2, 52, 89] methods have been noticeably applied in X-ray CT research. However, the large scale and the shift-variance of the statistical 3D X-ray CT cost function have limited the effectiveness of existing methods, motivating the researchers to develop algorithms that are less dependent on the large and shift-variant nature of the 3D CT system. In addition, the iterative algorithms that are favorable to modern massively parallelizable computing techniques are of increasing interest [95], whereas sequentially updating CD algorithms are more difficult to parallelize than simultaneously updating algorithms such as PCG, EM and OS.

Recently, variable-splitting (VS) techniques [46, 75, 94], accompanied by the method-of-multipliers framework [13], have gotten attention from the community because they can circumvent the shift-variance of the statistical CT cost function. Initial efforts on 2D CT [94] seemed promising, but the significant shift-variance of 3D CT system has remained a challenge [44, 73]. In addition, substantial memory requirements and parameter tuning remain an issue in VS techniques [75, 87].

Considering the large measurement data of 3D CT system, OS methods that use only a subset of the measurement data per iteration have been used widely since their first introduction in 1994 [52]. OS methods can be applied to any gradient-based iterative algorithms, such as OS-CD [68], OS-PCG [76], and OS-EM [52]. The VS-based algorithms can also accommodate OS methods [85]. The acceleration gain from such simple OS application to gradient-based algorithms has been found to be very impressive, and the commercial positron emission tomography (PET) and single-photon emission CT (SPECT) products adopted OS-EM algorithms around 1997. The first work of this thesis is modifying OS methods so that they can handle 3D helical cone-beam CT geometry in Chapter III. However, the existing OS methods are not fast enough to be used routinely for every clinical CT scan, and this dissertation focuses on developing faster algorithms using OS methods.

Among many variants of OS methods, this work focuses on the version that is based on separable quadratic surrogates (SQS) [2]. The corresponding OS-SQS methods require simple implementation and are massively parallelizable. Their convergence performance is also less dependent on the shift-variant property of statistical CT reconstruction. Considering these advantages, OS-SQS methods have served as a state-of-the-art approach in X-ray CT research until recently [3, 16, 32, 69, 77, 101]. However, they require further acceleration to be used routinely for clinical CT, and this dissertation proposes two main approaches for accelerating OS-SQS algorithms, one that uses novel optimization transfer (OT) techniques in Chapter IV, and one that combines with momentum methods such as well-known Nesterov's momentum methods [79,83] and their relaxed version [29], called OS-momentum algorithms, in Chapter V. To further accelerate OS-momentum, this thesis proposes new momentum methods in Chapter VI, called optimized gradient methods, which are twice as fast yet have remarkably simple implementations comparable to Nesterov's methods. The specific contribution of each proposed approach for accelerating OS-SQS methods is discussed in next section.

In addition to OS-based algorithms, one variant of block CD (BCD) algorithms, called axial BCD (ABCD), is investigated in Chapter VII. Sequential CD algorithms and simultaneous PCG or OS-SQS algorithms have a trade-off between fast convergence rate and efficient computation [22, 96], and the proposed ABCD in 3D X-ray CT is expected to well-balance between two aspects. The specific contribution of this ABCD algorithm is discussed in next section.

This dissertation is organized as follows. Chapter II reviews X-ray CT physics, statistical X-ray CT image reconstruction, and optimization algorithms. Chapter III presents the OS methods and improves them for 3D helical cone-beam X-ray CT. Chapter IV proposes accelerated optimization transfer techniques that are combined with the OS methods.

Chapter V suggests combining the OS and (relaxed) momentum techniques like Nesterov's momentum. Chapter VI presents new momentum techniques that are twice faster than Nesterov's momentum. Chapter VII shows the ABCD method which is specifically designed for 3D cone-beam X-ray CT. Chapter VIII offers conclusion and future work.

1.1 Contribution

This section describes the specific contributions of this dissertation, namely developing fast iterative algorithms for statistical 3D X-ray CT reconstruction.

In 3D helical CT geometries, we observed that conventional OS algorithms for PL and PWLS problems are unstable for large subset numbers as they did not consider their nonuniform sampling. Thus, Chapter III describes an improved OS algorithm that takes account of the nonuniform sampling and is more stable for helical CT. In addition, OS methods typically approach a limit-cycle looping around the optimum of the cost function, and Chapter III suggests a simple approach that averages sub-iterations at the last iteration to improve convergence, when the algorithm reaches a limit-cycle. This work was initially presented in [61] and then included as a part of [62].

The SQS method is an optimization transfer (OT) method that replaces the original cost function by a simple surrogate function [53, 67] with a monotonic descent property. We usually combine this SQS with the OS method yielding OS-SQS method, because it is simple, efficient and massively parallelizable. However, the OS-SQS method itself is not fast enough, and thus Chapter IV proposes three novel approaches that accelerate SQS method without losing the monotonicity. First, we construct surrogates with spatially *nonuniform* curvatures that provide spatially *nonuniform* step sizes to accelerate convergence, which is effective because the difference between the initial and final images are spatially nonuniform as discussed in [109]. In other words, the proposed spatially nonuniform (NU) SQS method encourages larger updates for voxels that are predicted to be farther from the optimal value. Section 4.2.1 provides a theoretical justification for the acceleration of NU method by analyzing the convergence rate of the SQS algorithm. Second, Chapter IV suggests a SQS method with bounded intervals (SQS-BI) that designs a surrogate function defined within a bounded interval that is known to include the minimizer of the cost function, which leads to a reduced surrogate curvature, inspired by [109]. Chapter IV further proposes quasi-separable quadratic surrogate (QSQS) methods that construct a surrogate with a tridiagonal Hessian rather than a diagonal Hessian for the original SQS. This approach provides a good trade-off between the convergence rate and computation time per iteration. The NU-SQS work was initially introduced in [57] and led to a main part of [62]. The SQS-BI work was discussed

in [56].

Momentum approaches such as a heavy-ball method [93] and Nesterov's methods [79, 83] have received wide attention in the optimization community for cleverly reusing the previous updates for acceleration without any computational overhead. Particularly, Nesterov's methods have been applied to X-ray CT [7, 19, 55] with promising results, but we observed that these algorithms do not show significant improvement in 3D X-ray CT, due to the large Lipschitz constant of its CT cost function (as discussed in [94]). Therefore, Chapter V proposes combining OS methods and Nesterov's momentum approaches, named as OS-momentum algorithms, that provide very fast initial acceleration. However, we experienced some unstable behavior of the proposed OS-momentum algorithms. To stabilize the proposed methods, Chapter V adapted a relaxation scheme that is developed for stochastic gradient methods with momentum in [29]. We investigated various choices of relaxation scheme to achieve both fast initial acceleration and stability. The initial work combining OS and momentum was discussed in [63, 64] and the additional relaxation was introduced in [58]. All these works were gathered and further investigated in [65].

To further accelerate OS-momentum algorithms, Chapter VI develops new momentum methods, called optimized gradient methods, which are twice as fast yet have remarkably simple implementations comparable to Nesterov's momentum methods. This new momentum approach has been initially studied by Drori and Teboulle [31], *numerically* showing that the certain first-order algorithms provide twice faster convergence rate than the state-of-the-art Nesterov's methods. However, the corresponding algorithm in [31] remained computationally undesirable and relied only on *numerical* analysis. The main contribution of Chapter VI is making the Drori and Teboulle's work practical implementation-wise and *analytically* showing that new proposed algorithms are twice as fast as Nesterov's methods. This work was introduced in [59], and its OS version has been investigated in [60].

Finally, Chapter VII suggests an ABCD algorithm that is specifically designed for 3D cone-beam X-ray CT geometries. This uses an axial block in a general BCD framework [49, 54], leading to both a fast convergence rate and an efficient implementation along with a separable footprint (SF) projector [70]. In BCD algorithms of X-ray CT, the coupling between voxels within the block becomes an issue considering the convergence rate and efficient implementation. Block iterative CD (B-ICD) [12] and group coordinate descent (GCD) [41] methods have been previously proposed in 2D CT problems considering the coupling effects. The ABCD algorithm can be viewed as an extension of them that leads to less coupling between voxels regarding the 3D CT geometry, and this is expected to provide an accelerated convergence rate comparable to ICD [96, 109] updating one voxel at a time, while the ABCD having more parallelization opportunity from updating a block of

voxels simultaneously. The ABCD algorithm works hand-in-hand with SF projector, and we provide a way to efficiently implement the ABCD algorithm. This ABCD approach was presented in [42].

CHAPTER II

Background

This chapter provides a background of X-ray CT image reconstruction and its statistical approach. Then, optimization methods for statistical X-ray CT reconstruction are discussed.

2.1 Background of X-ray CT image reconstruction

2.1.1 X-ray CT physics

X-ray CT images a distribution of X-ray attenuation coefficients $\mu(\vec{r})$ of an object, where $\vec{r}$ represents the spatial location. An X-ray CT system basically consists of a source and a detector array rotating around the patient (see Fig. 2.1). A patient is scanned along several different projection views. A "projection" here refers to a process of emitting X-ray photons from a source through the patient and recording the intensity of those photons that pass through the object.

Each projection can be viewed as a ray of X-ray photons for each pair of projection views and detector elements, where i is the index for each ray and N_d is the total number of rays. The following is Beer's law for the mono-energetic[1] X-ray CT projection process:

$$\bar{Y}_i = b_i \exp\left(- \int_{L_i} \mu(\vec{r}) dl \right) + r_i, \quad \text{for } i = 1, \cdots, N_d, \tag{2.1}$$

where $\bar{Y}_i$ is mean detected intensity, b_i is initial intensity of a beam, r_i is room background and mean scatter, and L_i is a line path of the ray through the object. Based on the Beer's law (2.1), the measured $\bar{Y}_i$ has the information of total attenuation encountered by the ith ray, *i.e.*, the line integral of $\mu(\vec{r})$ along the path L_i. In practice, we observe the *noisy* measured intensity Y_i of a beam, rather than the mean $\bar{Y}_i$ of the beam.

[1] In this thesis, we focus on the case where this (mono-energetic) Beer's law (2.1) holds by assuming that the measured data has been precorrected for the effects from a poly-energetic X-ray source and beam hardening [32].

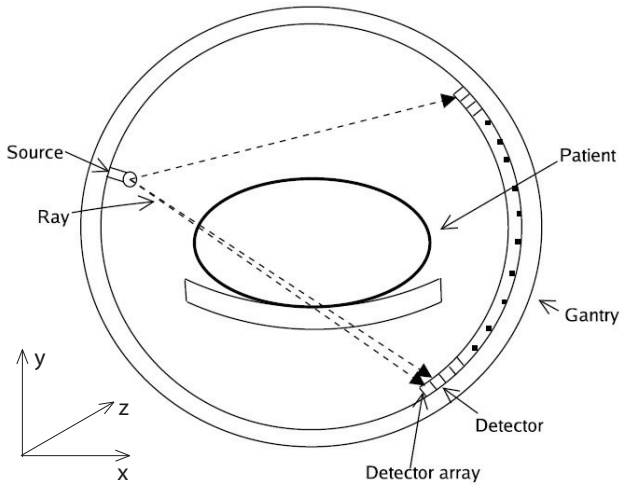


Figure 2.1: Schematic diagram of a X-ray CT scanner [100].

One X-ray CT scan provides N_d line integral values of $\mu(\vec{r})$ for each ray (see Fig. 2.2), which are equivalent to the following post-log data $\bar{y}_i$ of mean measurement $\bar{Y}_i$:

$$\bar{y}_i \triangleq \log\left(\frac{b_i}{\bar{Y}_i - r_i}\right) = \int_{L_i} \mu(\vec{r})dl, \quad \text{for } i = 1, \cdots, N_d. \tag{2.2}$$

Note that the *noisy* measured projection data y is available in practice rather than the mean projection data $\bar{y}$. The measured projection data y (or Y) is called a sinogram, and an example of sinogram data for 2D X-ray CT is shown in Fig. 2.3(b) with the orientation of projection views and detector elements.

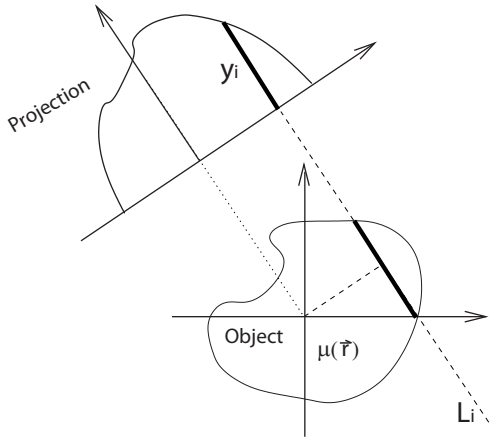
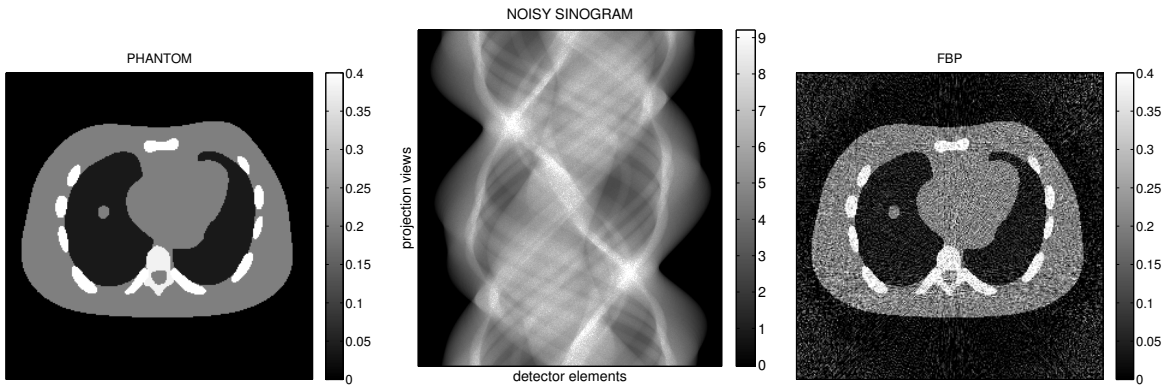


Figure 2.2: The measurement y_i is acquired by forward projecting $\mu(\vec{r})$ along a line path L_i.



(a) (b) (c)

Figure 2.3: Simulation of forward projection and FBP: (a) Phantom image, (b) noisy sinogram and (c) FBP image.

Since the scanning process of CT provides forward projected measurements, one might think that a simple back-projection of the data may reconstruct an image of the patient's body. However, a back-projection blurs the image with a point spread function $\frac{1}{||\vec{r}||}$. To overcome this blur, an (apodized) ramp filter is introduced to prefilter the sinogram. Then the filtered back-projection (FBP) algorithm reconstructs an image by back-projecting such filtered sinogram [34]. Fig. 2.3 illustrates the sinogram and the FBP reconstructed image for a simple phantom image.

2.1.2 Object parametrization

To process the reconstruction on a computer, it is necessary to discretize a continuous object $\mu(\vec{r})$ by a linear series expansion. We parametrize the object using a basis function $p_j(\vec{r})$ at each jth location as follows:

$$\mu(\vec{r}) \approx \sum_{j=1}^{N_p} x_j p_j(\vec{r}), \tag{2.3}$$

where x_j represents the coefficients of an object $\mu(\vec{r})$ in a discrete domain with the corresponding basis function. Here, the choice of basis function affects the image quality and the complexity of computation. Two popular choices of basis functions are voxels and blobs. Voxels are easy to compute since they are rectangular-shaped, but have a disadvantage that the voxels have a large side-lobe in the frequency domain, which may amplify errors of high frequency information of an object. On the other hand, smooth blobs are band-limited in the frequency domain, but computationally more complicated due to a larger footprint in

the projection domain. For the numerical results in this thesis, voxels are used as a basis function for their simplicity, but the proposed algorithms are applicable for any choice of basis functions.

The object parametrization (2.3) leads to a following discretization of a forward projection in (2.2) as:

$$y_i = \int_{L_i} \mu(\vec{r}) dl \approx \int_{L_i} \sum_{j=1}^{N_p} x_j p_j(\vec{r}) dl = \sum_{j=1}^{N_p} x_j \int_{L_i} p_j(\vec{r}) dl = \sum_{j=1}^{N_p} a_{ij} x_j \triangleq [Ax]_i, \tag{2.4}$$

where the footprint $a_{ij} \triangleq \int_{L_i} p_j(\vec{r}) dl$, depends on the model of a basis function $p_j(\vec{r})$ and a ray L_i. The matrix A is called X-ray CT system matrix, or simply system matrix or projector.

The size of the system matrix A is large in 3D CT geometry, and thus it is impractical to store it in fast random access memory (RAM), even if we exploit the sparse structure of A. One typical size of A in 3D helical CT scan is $N_p \times N_d \approx (2 \cdot 10^8) \times (3 \cdot 10^7) \approx 6 \cdot 10^{15}$, where the size of image x is $N_p = 512 \times 512 \times 109$ and that of measurement y is $N_d = 888 \times 32 \times 7146$ (the number of detector columns $\times$ detector rows $\times$ projection views). Considering that each ray intersects about two times the number of voxels of an axis along the transaxial plane, the sparsity of A can be approximated as $\frac{\sum_{j=1}^{N_p} I_{\{a_{ij}>0\}}}{N_p} \approx \frac{2 \times 512}{512 \times 512 \times 109} \approx 4 \cdot 10^{-5}$. However, even with this high sparse nature of A, the memory requirement for storing A remains large, about $4 \cdot 10^{-5} \times 6 \cdot 10^{15} \times 4$ Byte= 960 GByte for single-precision floating point. Instead of storing A, the matrix-vector multiplications Ax and $A'y$, called forward and back projection operations, are computed on the fly, which are the computational bottleneck of 3D X-ray CT. We discuss the details of efficient computation of projection operations for 3D cone-beam X-ray CT in Section 7.1.2.

2.2 Statistical image reconstruction

For low-dose scans, a conventional FBP reconstruction cannot construct a decent image due to increased noise (see Fig. 2.3(c)). Therefore, we use statistical modeling for improved image reconstruction.

2.2.1 Statistical modeling of X-ray CT scan

Several statistical models have been proposed for X-ray CT. The two simple noise models mostly used in practice are explained here. The pre-log observation data is often modeled

by a Poisson model of a discretized version of (2.1) [3]:

$$Y_i \sim \mathsf{Poisson}\big\{\bar{Y}_i\big\} = \mathsf{Poisson}\big\{b_i \, \mathrm{e}^{-[Ax]_i} + r_i\big\}, \quad \text{for } i = 1, \ldots, N_d, \tag{2.5}$$

where b_i and r_i are known nonnegative constants.

The post-log observation data $y_i = \log(b_i/(Y_i - r_i))$ in (2.2) is often modeled by a linear additive Gaussian noise model, which comes from quadratically approximating the negative likelihood of (2.5) [35]:

$$y_i = \bar{y}_i + \epsilon_i = [Ax]_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_i^2), \quad \text{for } i = 1, \ldots, N_d, \tag{2.6}$$

where the noise variance is $\sigma_i^2 = \frac{\bar{Y}_i}{(\bar{Y}_i - r_i)^2}$.

2.2.2 Statistical X-ray CT image reconstruction

Based on the previous section, we reconstruct a nonnegative image $x = (x_1, \ldots, x_{N_p}) \in \mathbb{R}_+^{N_p}$ from noisy measured transmission data $Y \in \mathbb{R}^{N_d}$ by minimizing either penalized likelihood (PL) [1,32,36] or penalized weighted least-squares (PWLS) [96,102,103] cost functions:

$$\hat{x} = \arg\min_{x \succeq 0} \Psi(x), \tag{2.7}$$

$$\Psi(x) \triangleq L(x) + R(x) = \sum_{i=1}^{N_d} h_i([Ax]_i) + \sum_{k=1}^{N_r} \psi_k([Cx]_k), \tag{2.8}$$

where $\hat{x}$ is a minimizer of $\Psi(x)$ subject to a nonnegativity constraint. The function $L(x)$ is a negative log-likelihood term (data-fit term) and $R(x)$ is a regularizer. The matrix $A \triangleq \{a_{ij}\}$ is a projection operator ($a_{ij} \geq 0$ for all i, j) where $[Ax]_i \triangleq \sum_{j=1}^{N_p} a_{ij} x_j$, and $C \triangleq \{c_{kj}\}$ is a finite differencing matrix considering 26 neighboring voxels in 3D image space.[2] The function $h_i(t)$ is selected based on the chosen statistics and physics:

- PL for pre-log data Y_i with Poisson model [1, 32, 36] in (2.5) uses:

$$h_i(t) = (b_i e^{-t} + r_i) - Y_i \log\big(b_i e^{-t} + r_i\big), \tag{2.9}$$

  where b_i is the blank scan factor and r_i is the mean number of background events. The function $h_i(\cdot)$ is nonconvex if $r_i \neq 0$ or convex otherwise. A shifted Poisson model [107] that partially accounts for electronic recorded noise can be used instead.

[2] Each row of C consists of a permutation of $(1, -1, 0, \ldots, 0) \in \mathbb{R}^{N_p}$ where the indices of the nonzero entries 1 and -1 corresponds to adjacent voxel locations in 3D image space.

- PWLS for post-log data $y_i = \log(b_i/(Y_i - r_i))$ with Gaussian model [96, 102, 103] in (2.6) uses a convex quadratic function:

$$h_i(t) = \frac{1}{2} w_i (t - y_i)^2, \tag{2.10}$$

where $w_i = (Y_i - r_i)^2 / Y_i$ provides statistical weighting. Then $L(x) = \frac{1}{2}||y - Ax||_W^2$ for PWLS with a diagonal matrix $W = \mathsf{diag}\{w_i\}$.

The function $\psi_k(t)$ is a (convex and typically nonquadratic) edge-preserving potential function:

$$\psi_k([Cx]_k) \triangleq \beta_k \psi([Cx]_k), \tag{2.11}$$

where spatial weighting β_k balances between the data-fit term and regularizer, and it can be designed to provide uniform resolution properties [43]. The function $\psi(t)$ can be chosen from many options, including the following:

- Hyperbola potential function [17]:

$$\psi(t) = \frac{\delta^2}{3} \left(\sqrt{1 + 3(t/\delta)^2} - 1 \right), \tag{2.12}$$

- Generalized Fair potential function[3] [33]:

$$\psi(t) = \frac{\delta^2}{b^3} \left(\frac{ab^2}{2} \left| \frac{t}{\delta} \right|^2 + b(b-a) \left| \frac{t}{\delta} \right| + (a-b) \log \left(1 + b \left| \frac{t}{\delta} \right| \right) \right). \tag{2.13}$$

These nonquadratic cost functions $\Psi(x)$ cannot be minimized analytically, and require many iterations of optimization algorithms to minimize the cost function. This process demands long computation time in X-ray CT, and thus our main goal is to develop optimization algorithms that provide fast convergence rate in X-ray CT image reconstruction with computational efficiency.

2.3 Optimization algorithms

This section discusses optimization algorithms that have been found useful for minimizing the X-ray CT cost function (2.8) efficiently and rapidly, and reviews specific algorithms such

[3] The potential function (2.13) is a generalized version of Fair potential function in [33], and the equation (2.13) reduces to the original when $a = 0$ and $b = 1$.

as optimization transfer (OT) techniques [53, 67], SQS methods [2], OS methods [2, 52], and momentum approaches [79, 83] that are used and extended in this dissertation.

2.3.1 Optimization algorithms in X-ray CT problem

Many iterative algorithms have been applied and developed for statistical X-ray CT problems. However, none of them can optimize the *large* and *shift-variant* statistical 3D X-ray CT cost function within short computation time. Iterative algorithms that can circumvent those two difficulties of the X-ray CT problem are likely to be effective in accelerating statistical X-ray CT reconstruction. In addition, considering the modern parallelizable computing resources, algorithms that are amenable to massive parallelization are preferable. Here we point out some optimization algorithms that have been extensively developed and used in X-ray CT research such as CD, PCG, OS-SQS, VS and momentum methods.

CD algorithms [109] (also known as Gauss-Siedel algorithms [47, p. 507]) and BCD algorithms [12, 41], update one or a group of voxels sequentially. These can converge (to within some tolerance) in few iterations but can require long computation times per iteration [22, 96]. Considering modern computing architectures, algorithms that update all voxels simultaneously and that are amenable to parallelization are desirable, such as OS-SQS [2] and PCG algorithms [38,45]. (Similarly, BCD may have more opportunity for parallelization than CD algorithms that sequentially update one voxel at a time.) However, those highly parallelizable algorithms require more iterations than CD algorithms [22, 96], and thus it is desirable to reduce the number of iterations needed to reach acceptable images, which is the main goal of this dissertation.

The parallelizable PCG algorithms [38] are widely used in general optimization algorithms (and X-ray CT problems), but the the shift-variance of the statistical 3D cost function makes it difficult to find an efficient and shift-invariant preconditioner that approximates a shift-variant Hessian of the cost function well. (Here the accuracy of the approximation of the Hessian is the key to the acceleration.) Recently, VS techniques [46, 75, 94] accompanied by the method-of-multipliers framework [13] have been introduced, dividing the problem into easier sub-problems relieving the shift-variant nature of the statistical CT problems. Even though the initial work on 2D CT [94] looked promising, the significant shift-variance of the 3D CT system made the VS approaches less effective [44, 73]. (VS approach was initially proposed along with PCG algorithms [75, 94], but recent works [74, 86] use other methods such as OT or duality approach to circumvent the shift-variant nature.) In addition, their increased memory in 3D CT and parameter tuning remain a problem. (A reduced memory version [75] is available but provides slow convergence.)

Highly parallelizable OS-SQS methods [2] are less dependent on both large-scale and

shift-variance of the cost function, and thus we focus on further accelerating these in this dissertation. OS methods [52] use only a subset of the measurement data per iteration for computational efficiency, and have been used widely in tomography problems. OS methods are usually combined with the SQS methods [2] because those are simple, efficient, and massively parallelizable. We provide more details in next Sections 2.3.2.2 and 2.3.3.

Momentum techniques [10, 79, 81, 83] that use previous update directions as momentum towards the optimum for acceleration with minimal computational overhead become popular in optimization community. These have been applied to X-ray CT [7,19,55], but we observed that the algorithms are not fast enough in 3D X-ray CT problem (as discussed in [94]), which we further accelerate by combining with OS methods in Chapter V. The details of momentum techniques are reviewed in Section 2.3.4.

2.3.2 Optimization transfer methods

The previous section discussed couple of optimization algorithms that are widely used or recently introduced in X-ray CT reconstruction. Here, we review a useful optimization transfer (OT) framework [53,67] that is widely applied in X-ray CT optimization algorithms. In general, any optimization algorithm has to determine a step size for the updating descent direction such as a negative gradient at each iteration. Exactly (or almost exactly) searching the optimal step size along the descent direction is expensive for a nonquadratic and large-scale X-ray CT cost function (2.8). Alternatively, we usually adapt an OT method that replaces a complicated cost function by a surrogate function at every iteration that is easier to minimize. This may not provide the optimal step size for a given updating descent direction, but improves computational efficiency while somewhat preserving the convergence rate.

In other words, when a cost function $\Psi(x)$ is difficult to minimize, an OT method replaces $\Psi(x)$ with a surrogate function $\phi^{(n)}(x)$ at the nth iteration for computational efficiency. This is also known as a majorization-minimization principle [91], and a comparison function [51]. OT methods include well-known expectation maximization (EM) algorithms [23,28], separable surrogate (SS) algorithms based on De Pierro's lemma [24,25,66] and surrogate algorithms using Lipschitz constants [10, 20].

The basic iteration of OT is

$$x^{(n+1)} = \arg\min_{x \succeq 0} \phi^{(n)}(x). \tag{2.14}$$

To monotonically decrease $\Psi(x)$, we design surrogate functions $\phi^{(n)}(x)$ that satisfy the fol-

lowing majorization conditions:

$$\Psi(x^{(n)}) = \phi^{(n)}(x^{(n)}), \quad \Psi(x) \leq \phi^{(n)}(x), \quad \forall\, x \in \mathbb{R}_+^{N_p}. \tag{2.15}$$

Constructing surrogates with smaller curvatures while satisfying condition (2.15) is the key to faster convergence in the OT methods [41]; Chapter IV investigates different ways to construct a surrogate that leads to both fast convergence rate and efficient computation.

The OT method has been used widely in tomography problems. De Pierro developed a SS approach in emission tomography [24, 25]. Quadratic surrogate (QS) functions have been derived for nonquadratic problems, enabling monotonic descent [1]. SQS algorithms combine SS and QS [2] and are the focus of Chapter IV. Partitioned SQS methods for multi-core processors have been proposed for separating the image domain by the number of processors and updating each of them separately while preserving monotonicity [99].

We use this OT method implicitly and explicitly for the efficient implementation for all optimization algorithms used in this dissertation. We propose accelerating iterative algorithms in Chapter IV that extend SQS algorithms and converge faster than the standard SQS. The next two sections show two examples of OT methods.

2.3.2.1 Optimization transfer using the Lipschitz constant

The cost function $\Psi(x)$ that is continuously differentiable with Lipschitz constant $\mathcal{L}$, *i.e.*,

$$||\nabla\Psi(x) - \nabla\Psi(z)|| \leq \mathcal{L}||x - z||, \quad \forall x, z \in \mathbb{R}_+^{N_p}$$

can be majorized at the nth iteration as:

$$\Psi(x) \leq \phi_{\mathcal{L}}^{(n)}(x) \triangleq \Psi(x^{(n)}) + \nabla\Psi(x^{(n)})'(x - x^{(n)}) + \frac{\mathcal{L}}{2}||x - x^{(n)}||^2, \quad \forall x \in \mathbb{R}_+^{N_p}, \tag{2.16}$$

which satisfies (2.15). This construction leads to the following algorithm in Table 2.1, where the notation $[\cdot]_+$ enforces a nonnegativity constraint.

1: Initialize $x^{(0)}$.
2: for $n = 0, 1, \cdots, N-1$
3: $\quad x^{(n+1)} = \arg\min_{x \succeq 0} \phi_{\mathcal{L}}^{(n)}(x) = \left[x^{(n)} - \frac{1}{\mathcal{L}}\nabla\Psi(x^{(n)})\right]_+$
4: end

Table 2.1: OT methods using the Lipschitz constant $\mathcal{L}$

Table 2.1 is simple and provides monotonic descent updates with the following conver-

gence rate [10]:

$$\Psi(x^{(n)}) - \Psi(\hat{x}) \leq \frac{\mathcal{L}||x^{(0)} - \hat{x}||^2}{2n}. \tag{2.17}$$

Note that the above "global" convergence rate inequality holds for all iterations (n), unlike the root-convergence factor [91] (in Section 4.2.1) that quantifies the "asymptotic" convergence behavior. Considering that we are interested in developing iterative algorithms that provide fast initial convergence rate rather than fast asymptotic rate, we focus on studying the "global" convergence behavior like (2.17) throughout this dissertation.

In 3D X-ray CT, computing the (smallest) Lipschitz constant $\mathcal{L}$ is expensive, so the efficient SQS methods described next are usually preferred instead.

2.3.2.2 Separable quadratic surrogates methods

SQS methods [2] construct the following surrogate function with a diagonal Hessian matrix $D^{(n)}$ at the nth iteration:

$$\Psi(x) \leq \phi_{\text{SQS}}^{(n)}(x) \triangleq \Psi(x^{(n)}) + \nabla\Psi(x^{(n)})'(x - x^{(n)}) + \frac{1}{2}||x - x^{(n)}||_{D^{(n)}}^2, \quad \forall x \in \mathbb{R}_+^{N_p}, \tag{2.18}$$

which satisfies (2.15). In PWLS reconstruction, for example, the standard SQS algorithm [2] uses the following (precomputed) diagonal majorizing matrix:

$$D \triangleq \mathsf{diag}\Big\{ A'WA\mathbf{1} + |C|' \,\mathsf{diag}\Big\{ \ddot{\psi}_k(0) \Big\} |C|\mathbf{1} \Big\} \tag{2.19}$$

using the maximum curvature $\ddot{\psi}_k(0) = \max_t \ddot{\psi}_k(t)$ [1], where $W = \mathsf{diag}\{w_i\} \in \mathbb{R}_+^{N_d \times N_d}$ and $|C| \triangleq \{|c_{kj}|\} \in \mathbb{R}_+^{N_r \times N_p}$, and the vector $\mathbf{1} \in \mathbb{R}^{N_p}$ consists of N_p ones. The detailed derivation of D in (2.19) and its extension using an iteration-dependent $D^{(n)}$ for acceleration are discussed in Chapter IV.

Table 2.2 gives the outline of the computationally efficient (and massively parallelizable) SQS algorithm using the precomputed diagonal D, where the operation $[\cdot]_+$ enforces a nonnegativity constraint by a (simple) element-wise clipping that replaces negative element values to zero. The convergence rate of the SQS methods (that extends (2.17)) and its acceleration are discussed in Chapter IV.

Both OT methods using Lipschitz constant and SQS methods in Tables 2.1 and 2.2 are preferable implementation-wise, since they only require either $\frac{1}{\mathcal{L}}$ or D^{-1} simple operation per iteration, in addition to the gradient computation $\nabla\Psi(x)$. However, none of them lead to fast convergence due to large $\mathcal{L}$ (or $D^{(n)}$) in (2.17). Therefore, these are usually combined with OS methods in tomography problems for accelerated (initial) convergence rates, which we review in next section.

1: Initialize $x^{(0)}$ and compute D.
2: for $n = 0, 1, \cdots, N-1$
3: $\quad x^{(n+1)} = \arg\min_{x \succeq 0} \phi^{(n)}_{\mathrm{SQS}}(x) = \left[x^{(n)} - D^{-1}\nabla\Psi(x^{(n)})\right]_+$
4: end

Table 2.2: SQS methods

2.3.3 Ordered subsets methods

X-ray CT iterative reconstruction requires both forward and back projection operations Ax and $A'y$ on the fly [21, 70] due to their large-scale in 3D, and thus computation of the gradient $\nabla\Psi(x)$ requiring two projections is very expensive. OS methods [52] accelerate gradient-based algorithms by grouping the projection views into M subsets (approximately evenly) and using only the subset of measured data to approximate the exact gradient of the cost function.

In PWLS reconstruction, for example, OS methods define the subset-based cost function:

$$\Psi_m(x) \triangleq \frac{1}{2}||y_m - A_m x||^2_{W_m} + \frac{1}{M}R(x) \tag{2.20}$$

for $m = 0, 1, \cdots, M-1$, where $\Psi(x) = \sum_{m=0}^{M-1} \Psi_m(x)$ and the matrices y_m, A_m, W_m are sub-matrices of y, A, $W = \mathsf{diag}\{w_i\}$ for the mth subset, and rely on the following "subset balance" approximation [2, 52]:

$$\nabla\Psi(x) \approx M\nabla\Psi_0(x) \approx \cdots \approx M\nabla\Psi_{M-1}(x). \tag{2.21}$$

Using (2.21), OS methods provide initial acceleration of about the factor of the number of subsets M by replacing $\nabla\Psi(x)$ in Table 2.2 into $M\nabla\Psi_m(x)$. Table 2.3 shows the OS version of the SQS method in Table 2.2.

1: Initialize $x^{(0)}$ and compute D.
2: for $n = 0, 1, \cdots, N-1$
3: for $m = 0, 1, \cdots, M-1$
4: $\quad k = nM + m$
5: $\quad x^{(\frac{k+1}{M})} = \left[x^{(\frac{k}{M})} - D^{-1}M\nabla\Psi_m(x^{(\frac{k}{M})})\right]_+$
6: end
7: end

Table 2.3: OS-SQS methods

We count one iteration after all M sub-iterations are performed considering the use of projection operators A and A' per iteration, and expect to have initial acceleration of $O\left(\frac{1}{nM}\right)$. (Using large M can slow down the algorithm in run time due to the regularizer computation [18].) OS algorithms approach a limit-cycle because the assumption (2.21) breaks near the optimum [71]. OS algorithms can be modified to converge to the optimum with some loss of acceleration in early iterations [4, 5].

2.3.4 Momentum methods

Momentum approaches are optimization methods that use previous updating directions in addition to the current one to provide acceleration toward the optimum. A heavy-ball method [93] and Nesterov's first-order methods [79, 83] are representative of the momentum approaches, and we focus on Nesterov's work here. Nesterov published a fast first-order method using the difference between two previous iterates as a momentum approach for smooth functions[4] in [79], and it was extended later for nonsmooth functions in [10], which is one of the state-of-the-art methods in image restoration [9]. In [83], Nesterov also proposed a new formulation of momentum using all previous iterates, which he also extended for nonsmooth functions [84].

Nesterov's two algorithms [79, 83] have been used widely for various optimization problems. This section briefly reviews them. Both [79] and [83] use an optimization transfer update described in Table 2.1. Then the algorithm is accelerated using previous iterates as shown in Tables 2.4 and 2.5 [79, 83]. We use the choice of parameters suggested in [104] for the algorithm in Table 2.5, which provides faster convergence than the original choice in [83].

1: Initialize $x^{(0)} = z^{(0)},\ t_0 = 1$
2: for $n = 0, 1, 2, \cdots$
3: $\quad t_{n+1} = \left(1 + \sqrt{1 + 4t_n^2}\right)/2$
4: $\quad x^{(n+1)} = \left[z^{(n)} - \frac{1}{\mathcal{L}}\nabla\Psi(z^{(n)})\right]_+$
5: $\quad z^{(n+1)} = x^{(n+1)} + \frac{t_n - 1}{t_{n+1}}(x^{(n+1)} - x^{(n)})$
6: end

Table 2.4: Nesterov's momentum method (1983) in [79]

The sequences $\left\{x^{(n)}\right\}$ generated by both algorithms have been proven to have the follow-

[4] A smooth function $f(x)$ is continuously differentiable with Lipschitz continuous gradient $\mathcal{L}$ satisfying $||\nabla f(x) - \nabla f(z)|| \leq \mathcal{L}||x - z||$ for all $x, z \in \mathbb{R}^{N_p}$.

1: Initialize $x^{(0)} = v^{(0)} = z^{(0)},\ t_0 = 1$

2: for $n = 0, 1, 2, \cdots$

3: $\quad t_{n+1} = \left(1 + \sqrt{1 + 4t_n^2}\right)/2$

4: $\quad x^{(n+1)} = \left[z^{(n)} - \frac{1}{\mathcal{L}} \nabla \Psi(z^{(n)})\right]_+$

5: $\quad v^{(n+1)} = \left[z^{(0)} - \frac{1}{\mathcal{L}} \sum_{k=0}^{n} t_k \nabla \Psi(z^{(k)})\right]_+$

6: $\quad z^{(n+1)} = \left(1 - \frac{1}{t_{n+1}}\right) x^{(n+1)} + \frac{1}{t_{n+1}} v^{(n+1)}$

7: end

Table 2.5: Nesterov's momentum method (2005) in [83]

ing convergence rate [79, 83]:

$$\Psi(x^{(n)}) - \Psi(\hat{x}) \leq \frac{2\mathcal{L}||x^{(0)} - \hat{x}||^2}{n^2}, \tag{2.22}$$

if $\Psi(x)$ is convex and has $\mathcal{L}$-Lipschitz gradient. This $O\left(\frac{1}{n^2}\right)$ rate is promising since ordinary optimization transfer provides only $O\left(\frac{1}{n}\right)$ rate in (2.17). However, the large Lipschitz constant $\mathcal{L}$ in CT problem causes slow convergence even with the $O\left(\frac{1}{n^2}\right)$ rate. In Chapter V, we illustrate that the rate $O\left(\frac{1}{n^2}\right)$ plays an important role for the very fast convergence rate of the proposed momentum methods combined with OS methods.

CHAPTER III

Ordered subsets for 3D cone-beam X-ray CT

This chapter describes some simple modifications of ordered subsets (OS) algorithms that improve their effectiveness for 3D CT. We first review the OS framework that is widely used in tomography problems for fast initial convergence rate. OS methods can accelerate algorithms by the number of subsets in early iterations by using a subset of the measured data for each subset update. OS algorithms are most effective when a properly scaled gradient of each subset data-fit term approximates the gradient of the full data-fidelity term, and then the algorithm can accelerate convergence by a factor of the number of subsets. However, standard OS algorithms usually approach a limit-cycle where the sub-iterations loop around the optimal point. OS algorithms can be modified so that they converge by introducing relaxation [4], reducing the number of subsets, or by using incremental optimization transfer methods [5]. Unfortunately, such methods converge more slowly than ordinary OS algorithms in early iterations. Therefore, we investigated averaging the sub-iterations when the algorithm reaches a limit-cycle, which improves image quality without slowing convergence. (There was a preliminary simulation study of this approach in [6].)

In cone-beam CT, the user must define a region-of-interest (ROI) along the axial (z) direction for image reconstruction (see Fig. 3.1). Model-based reconstruction methods for cone-beam CT should estimate many voxels outside the ROI, because parts of each patient usually lie outside the ROI yet contribute to some measurements. However, accurately estimating non-ROI voxels is difficult since they are incompletely sampled, which is called the "long-object problem" [26]. Reconstructing the non-ROI voxels adequately is important, as they may impact the estimates within the ROI. Unfortunately in OS algorithms, the sampling of these extra slices leads to very imbalanced subsets particularly for large numbers of subsets, which can destabilize OS algorithms outside the ROI. Here we propose an improved OS algorithm that is more stable for 3D helical CT by defining better scaling factors for the subset-based gradient [61].

This work was published in a conference proceedings [61], and is included as a part of a journal paper [62].

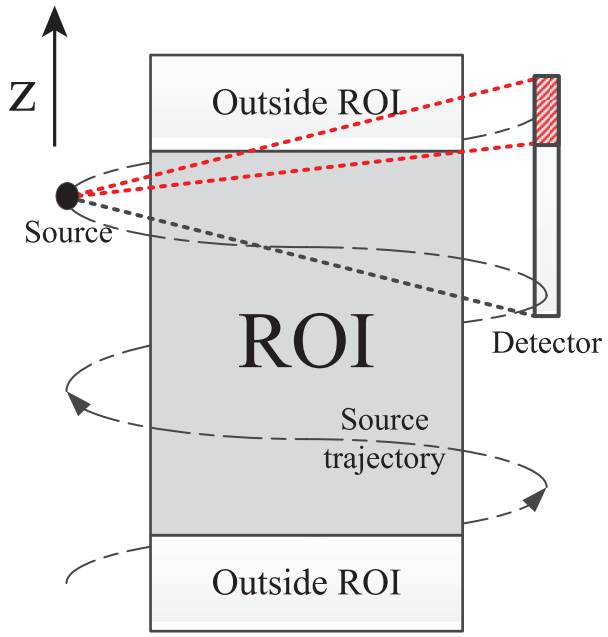


Figure 3.1: Diagram of helical CT geometry. The (red) dashed region indicates the detector rows that measure data with contributions from voxels both within and outside the ROI.

3.1 Ordered subsets (OS) methods

Here, we focus on a simple diagonally preconditioned algorithm, the SQS method, (see Sections 2.3.2.2 and 4.1) that generates an image sequence $\{x^{(n)}\}$ as follows:

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{1}{d_j^{(n)}} \left(\frac{\partial}{\partial x_j} L(x^{(n)}) + \frac{\partial}{\partial x_j} R(x^{(n)}) \right) \right]_+, \tag{3.1}$$

where $d_j^{(n)}$ can be derived using SQS method (see Section 4.1), and $[\cdot]_+$ enforces a nonnegativity constraint. Then, an OS algorithm (with M subsets) for accelerating the update (3.1) has the following mth sub-iteration within the nth iteration:

$$x_j^{(n+\frac{m+1}{M})} = \left[x_j^{(n+\frac{m}{M})} - \frac{1}{d_j^{(n)}} \left(\hat{\gamma}_j^{(n+\frac{m}{M})} \frac{\partial}{\partial x_j} L_m \left(x^{(n+\frac{m}{M})} \right) + \frac{\partial}{\partial x_j} R \left(x^{(n+\frac{m}{M})} \right) \right) \right]_+, \tag{3.2}$$

where $\hat{\gamma}_j^{(n+\frac{m}{M})}$ scales the gradient of a subset data-fit term

$$L_m(x) = \sum_{i \in S_m} h_i\left([Ax]_i\right), \tag{3.3}$$

and S_m consists of projection views in mth subset for $m = 0, 1, \cdots, M-1$. We count one iteration when all M subsets are used once, since the projection A used for computing data-fit gradients is the dominant operation.

If we use many subsets to attempt a big acceleration in the OS algorithm, some issues arise. The increased computation for the gradient of regularizer in (3.2) can become a bottleneck (this can be relieved by [18]). In addition, having less measured data in each subset will likely break the subset balance condition

$$\nabla L_0(x) \approx \nabla L_1(x) \approx \cdots \approx \nabla L_{M-1}(x). \tag{3.4}$$

The update in (3.2) would accelerate the SQS algorithm by exactly M if the scaling factor $\hat{\gamma}_j^{(n+\frac{m}{M})}$ satisfied the condition:

$$\hat{\gamma}_j^{(n+\frac{m}{M})} = \frac{\frac{\partial}{\partial x_j} L\left(x^{(n+\frac{m}{M})}\right)}{\frac{\partial}{\partial x_j} L_m\left(x^{(n+\frac{m}{M})}\right)}. \tag{3.5}$$

It would be impractical to compute this factor exactly, so the conventional OS approach is to simply use the constant $\gamma = M$. This "approximation" often works well in the early iterations when the subsets are suitably balanced, and for a small number of subsets. But in general, the errors caused by the differences between $\hat{\gamma}_j^{(n+\frac{m}{M})}$ and a constant scaling factor γ cause two problems in OS methods. First, the choice $\gamma = M$ causes instability in OS methods in a helical cone-beam CT that has limited projection views outside the ROI, leading to very imbalanced subsets. Therefore, Section 3.2 proposes an alternative choice γ_j that better stabilizes OS for helical CT. Second, even with γ replaced by γ_j, OS methods approach a limit-cycle that loops around the optimal point within sub-iterations [4]. Section 3.3 considers a simple averaging approach that reduces this problem.

3.2 Scaling factor for 3D helical geometry

The constant scaling factor $\gamma = M$ used in the ordinary regularized OS algorithm is reasonable when all the voxels are sampled uniformly by the projection views in all the subsets. But in geometries like helical CT, the voxels are nonuniformly sampled. In particular, voxels outside the ROI are sampled by fewer projection views than voxels within the ROI

(see Fig. 3.1). So some subsets make no contribution to such voxels, *i.e.*, subsets are very imbalanced. We propose to use a voxel-based scaling factor γ_j that considers the nonuniform sampling, rather than a constant factor γ.

After investigating several candidates, we focused on the following scaling factor that is expected to count the effective number of subsets for each voxel:

$$\gamma_j = \sum_{m=0}^{M-1} I_{\left\{\sum_{i \in S_m} \breve{c}_i^{(n)} a_{ij} \left(\sum_{l=1}^{N_p} a_{il}\right)\right\}}, \tag{3.6}$$

where $I_B = 1$ if B is true or 0 otherwise.[1] As expected, $\gamma_j < M$ for voxels outside the ROI and $\gamma_j = M$ for voxels within the ROI. We store (3.6) as a short integer for each voxel outside the ROI only, so it does not require very much memory.

We evaluated the OS algorithm with the proposed scaling factors (3.6) using the GE performance phantom (GEPP). Fig. 3.2 shows that the OS algorithm using the proposed scaling factors (3.6) leads to more stable reconstruction than the ordinary OS approach, which diverges outside the ROI. The instability seen with the ordinary OS approach may also degrade image quality within the ROI as seen by the noise standard deviations in Fig. 3.2. The results in Fig. 3.3 further show that the ordinary OS algorithm exhibits more variations within the ROI due to the instability outside ROI, whereas the proposed OS algorithm is robust.

3.3 Averaging sub-iterations

Although the new scaling factors (3.6) stabilize OS in helical CT and reduce artifacts, the final noise level is still worse than a convergent algorithm (see Fig. 3.2 and 3.3) because any OS method with constant scaling factors will not converge [71]. This section discusses one practical method that can reduce noise without affecting the convergence rate. This approach helps the OS algorithm come closer to the converged image, reducing the undesirable noise in images reconstructed using OS algorithms with large M.

To ensure convergence, the incremental optimization transfer (IOT) method [5] was proposed, which involves a form of averaging, but the greatly increased memory space required has prevented its application in 3D X-ray CT. As a practical alternative, we investigated an approach where the final image is formed by averaging all of the sub-iterations at the final iteration n_{end} of the OS algorithm (after it approaches its limit cycle). A memory-efficient

[1] The scaling factor (3.6) is particularly chosen so that the SQS method (in Section 4.1) will have small computation overhead as it can be computed simultaneously with the precomputation of the initial data-fit denominator (4.10). The notation $\breve{c}_i^{(n)}$ is explained in Section 4.1.

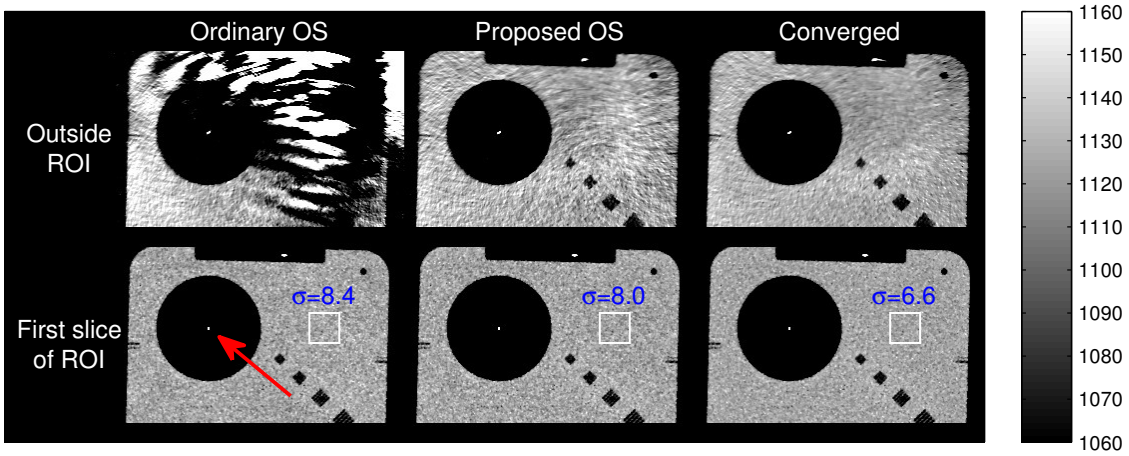


Figure 3.2: Effect of gradient scaling in regularized OS-SQS algorithm with GE performance phantom (GEPP) in helical CT: Each image is reconstructed after running 20 iterations of OS algorithm with 328 subsets, using ordinary and proposed scaling approaches. Standard deviation σ of a uniform region (in white box) is computed for comparison. We compute full-width half maximum (FWHM) of a tungsten wire (red arrow) to measure the resolution. (The result of a convergent algorithm is shown for reference. Images are cropped for better visualization.)

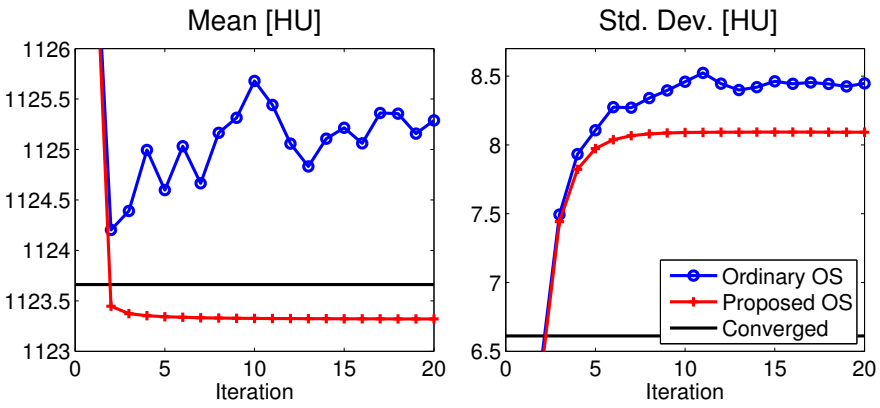


Figure 3.3: GE performance phantom: mean and standard deviation within a uniform region in the first slice of the ROI (see Fig. 3.2) vs. iteration, showing the instability of ordinary OS approach with 328 subsets, compared with the proposed OS approach. Also shown is the result from a converged image $x^{(\infty)}$ generated from several iterations of a convergent algorithm.

implementation of this approach uses a recursive in-place calculation:

$$\bar{x}^{\left(\frac{m+1}{M}\right)} = \frac{m}{m+1}\bar{x}^{\left(\frac{m}{M}\right)} + \frac{1}{m+1}x^{\left(n_{\text{end}}-1+\frac{m+1}{M}\right)}, \tag{3.7}$$

where $\bar{x}^{(0)}$ is an initial zero image, and $\bar{x}^{(1)}$ is the final averaged image. There was a preliminary simulation investigation of averaging the final iteration in [6], and we applied the averaging technique to CT scans here. In Table 3.1, we investigated this averaging method using a scan of the GEPP phantom and quantified the noise and resolution properties (as described in Fig. 3.2), and evaluated root mean square difference (RMSD) between current and converged image within the ROI:

$$\text{RMSD} \triangleq \frac{\left|\left|x_{\text{ROI}}^{(n)} - x_{\text{ROI}}^{(\infty)}\right|\right|_2}{\sqrt{N_{p,\text{ROI}}}} \text{ [HU]}, \tag{3.8}$$

where $N_{p,\text{ROI}}$ is the number of voxels in the ROI. Table 3.1 shows that the averaging technique successfully reduces the noise and RMSD.

	Smoothed FBP	OS-SQS(328)		Conv.
		w/o averaging	w/ averaging	
Mean [HU]	1127.7	1123.3	1123.8	1123.7
Std. Dev. [HU]	2.3	8.0	7.2	6.6
FWHM [mm]	1.4	0.7	0.7	0.7
RMSD [HU]	9.4	3.4	0.8	·

Table 3.1: GE performance phantom: Noise, resolution and RMSD behavior of OS-SQS(328 subsets) after 20 iterations followed by averaging.

3.4 Conclusion

We reviewed the OS algorithm and provided some simple modifications that were found to be helpful for 3D cone-beam CT reconstruction. We have refined the OS algorithm to handle a nonuniformly sampled helical CT geometry, which leads to more stabilized reconstruction. We have also adapted an averaging sub-iterations approach at the last iteration to reduce noise from the OS method.

CHAPTER IV

Accelerated optimization transfer methods

This chapter discusses acceleration in optimization transfer methods, particularly separable quadratic surrogates (SQS). SQS method is simple and massively parallelizable due to the *diagonal* Hessian of the surrogate, and can be easily combined with OS framework yielding OS-SQS method with fast initial convergence rate. But as it would benefit further acceleration, we carefully review SQS method and propose three novel approaches that extend SQS method to accelerate the convergence rate based on a careful derivation. First, we analyze the convergence rate of a standard SQS, and develop a spatially nonuniform SQS (NU-SQS) algorithm that provides larger step sizes for the voxels that need more updates. Second, with a knowledge of a bounded interval (BI) that includes the minimizer $\hat{x}$, we construct a tighter surrogate than the standard one, which we call SQS with BI (SQS-BI). Last, we generalize SQS method to construct a surrogate with a tridiagonal matrix that is tighter than the standard SQS, exploiting the fact that inverting a tridiagonal matrix can be done efficiently.

Each proposed idea in Section 4.2 and 4.3.1 was published in a conference proceedings [56, 57], and the former one combined with Chapter III lead to a journal paper [62].

4.1 Separable quadratic surrogates (SQS) methods

We first construct a quadratic surrogate at the nth iteration for the nonquadratic cost function in (2.8):

$$\Psi(x) = L(x) + R(x) \leq Q_L^{(n)}(x) + Q_R^{(n)}(x), \tag{4.1}$$

where $Q_L^{(n)}(x)$ and $Q_R^{(n)}(x)$ are quadratic surrogates for $L(x)$ and $R(x)$. Based on (2.8), the quadratic surrogate for $L(x)$ has the form:

$$Q_L^{(n)}(x) = \sum_{i=1}^{N_d} q_i^{(n)}([Ax]_i), \tag{4.2}$$

$$q_i^{(n)}(t) \triangleq h_i(t_i^{(n)}) + \dot{h}_i(t_i^{(n)})(t - t_i^{(n)}) + \frac{\breve{c}_i^{(n)}}{2}(t - t_i^{(n)})^2, \tag{4.3}$$

where $t_i^{(n)} \triangleq [Ax^{(n)}]_i$, and $\breve{c}_i^{(n)} = \max\left\{\breve{c}_i^{(n)}, \eta\right\}$ is the curvature of $q_i^{(n)}(t)$ for some small positive value η that ensures the curvature $\breve{c}_i^{(n)}$ positive [1]. In PWLS problem, $h_i(\cdot)$ is quadratic already, so $q_i^{(n)}(t) = h_i(t)$. The quadratic surrogate $Q_R^{(n)}(x)$ for $R(x)$ is defined similarly.

We choose curvatures $\left\{\breve{c}_i^{(n)}\right\}$ that satisfy the monotonicity conditions in (2.15). For PL, the smallest curvatures:

$$\breve{c}_i^{(n)} \triangleq \begin{cases} \left[2\frac{h_i(0) - h_i(t_i^{(n)}) + t_i^{(n)}\dot{h}_i(t_i^{(n)})}{[t_i^{(n)}]^2}\right]_+, & t_i^{(n)} > 0, \\ \left[\ddot{h}_i(0)\right]_+, & t_i^{(n)} = 0, \end{cases} \tag{4.4}$$

where $[t]_+ = \max\{t, 0\}$, called "optimal curvatures," lead to the fastest convergence rate but require an extra back-projection each iteration for nonquadratic problems [1]. Alternatively, we may use "maximum curvatures":

$$\breve{c}_i \triangleq \max_{t \geq 0} \ddot{h}_i(t) \tag{4.5}$$

that we can precompute before the first iteration [1].

Next, we generate a separable surrogate of the quadratic surrogate. For completeness, we repeat De Pierro's argument in [2]. We first rewrite forward projection $[Ax]_i$ as follows:

$$[Ax]_i = \sum_{j=1}^{N_p} a_{ij}x_j = \sum_{\substack{j=1 \\ a_{ij} \neq 0}}^{N_p} \pi_{ij}^{(n)} \left(\frac{a_{ij}}{\pi_{ij}^{(n)}}(x_j - x_j^{(n)}) + [Ax^{(n)}]_i\right), \tag{4.6}$$

where a nonnegative real number $\pi_{ij}^{(n)}$ is zero only if a_{ij} is zero for all i, j, and satisfies $\sum_{j=1}^{N_p} \pi_{ij}^{(n)} = 1$ for all i. Using the convexity of $q_i^{(n)}(\cdot)$ and the convexity inequality yields

$$q_i^{(n)}([Ax]_i) \leq \sum_{\substack{j=1 \\ a_{ij} \neq 0}}^{N_p} \pi_{ij}^{(n)} q_i^{(n)} \left(\frac{a_{ij}}{\pi_{ij}^{(n)}}(x_j - x_j^{(n)}) + [Ax^{(n)}]_i\right). \tag{4.7}$$

Thus we have the following SQS $\phi_L^{(n)}(x)$ (with a diagonal Hessian) for the data-fit term $L(x)$:

$$L(x) \le Q_L^{(n)}(x) \le \phi_L^{(n)}(x) \triangleq \sum_{j=1}^{N_p} \phi_{L,j}^{(n)}(x_j), \tag{4.8}$$

$$\phi_{L,j}^{(n)}(x_j) \triangleq \sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{N_d} \pi_{ij}^{(n)} q_i^{(n)} \left(\frac{a_{ij}}{\pi_{ij}^{(n)}} (x_j - x_j^{(n)}) + [Ax^{(n)}]_i \right). \tag{4.9}$$

The second derivative (curvature) of the surrogate $\phi_{L,j}^{(n)}(x_j)$ is

$$d_j^{L,(n)} \triangleq \frac{\partial^2}{\partial x_j^2} \phi_{L,j}^{(n)}(x_j) = \sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{N_d} \breve{c}_i^{(n)} \frac{a_{ij}^2}{\pi_{ij}^{(n)}}. \tag{4.10}$$

We can define a SQS $\phi_{R,j}^{(n)}(x_j)$ for the regularizer similarly, and it has the curvature:

$$d_j^{R,(n)} \triangleq \frac{\partial^2}{\partial x_j^2} \phi_{R,j}^{(n)}(x_j) = \sum_{\substack{k=1 \\ c_{kj} \neq 0}}^{N_r} \ddot{\psi}_k(0) \frac{c_{kj}^2}{\pi_{kj}^{(n)}}, \tag{4.11}$$

where $\pi_{kj}^{(n)}$ have similar constraints as $\pi_{ij}^{(n)}$, $\ddot{\psi}_k(0) = \max_t \ddot{\psi}_k(t)$ for maximum curvature [2], or $\ddot{\psi}_k(0)$ can be replaced by $\dot{\psi}_k\left([Cx^{(n)}]_k\right)/[Cx^{(n)}]_k$ for Huber's optimal curvature [51, Lemma 8.3, p.184].

Combining the surrogates for the data-fit term and regularizer and minimizing it in (2.14) leads to the following separable quadratic surrogate (SQS) method [2] that updates all voxels simultaneously with a "denominator" $d_j^{(n)} \triangleq d_j^{L,(n)} + d_j^{R,(n)}$:

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{1}{d_j^{(n)}} \frac{\partial}{\partial x_j} \Psi(x^{(n)}) \right]_+, \tag{4.12}$$

where a clipping $[\cdot]_+$ enforces the nonnegativity constraint. (Fig. 4.1 provides an illustration of the SQS update (4.12) on a 2D contour example, where it constructs a SQS surrogate $\phi^{(n)} \triangleq \phi_L^{(n)}(x) + \phi_R^{(n)}(x)$ at the nth iteration with a diagonal Hessian $D^{(n)} \triangleq \mathsf{diag}\left\{d_j^{(n)}\right\}$ and minimizes the surrogate to generate the next iterate $x^{(n+1)}$ as (4.12).) This SQS decreases the cost function $\Psi(x)$ monotonically, and it converges based on the proof in [53]. If $\Psi(x)$ is convex, a sequence $\left\{x^{(n)}\right\}$ converges to $x^{(\infty)}$ that is a global minimizer $\hat{x}$. Otherwise, $\left\{x^{(n)}\right\}$ converges to a local minimizer $x^{(\infty)}$ which may or may not be a global minimizer $\hat{x}$ depending on the initial image $x^{(0)}$.

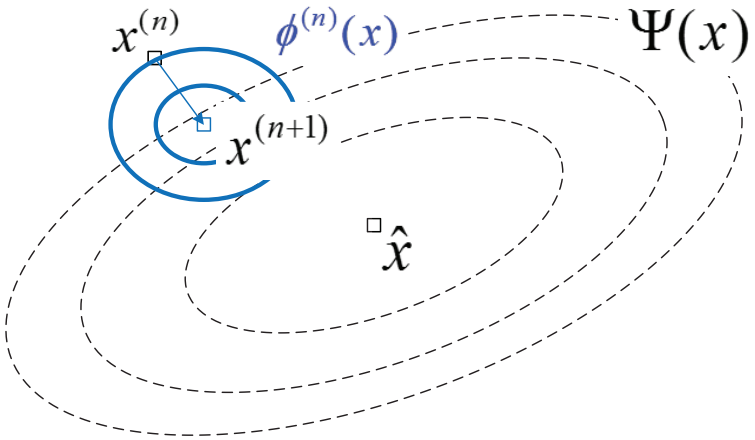


Figure 4.1: 2D illustration of SQS method: SQS methods construct a SQS surrogate $\phi^{(n)}$ and update to next iterate $x^{(n+1)}$ by minimizing the surrogate. Note that the shape of SQS surrogate $\phi^{(n)}$ is aligned to coordinate axes (for efficient implementation as in (4.12)), but not to the contour of the cost function $\Psi(x)$.

The implementation and convergence rate of SQS depend on the choice of $\pi_{ij}^{(n)}$. A general form for $\pi_{ij}^{(n)}$ is

$$\pi_{ij}^{(n)} \triangleq \frac{\lambda_{ij}^{(n)}}{\sum_{\substack{l=1 \\ a_{il} \neq 0}}^{N_p} \lambda_{il}^{(n)}}, \tag{4.13}$$

where a nonnegative real number $\lambda_{ij}^{(n)}$ is zero only if a_{ij} is zero. Then (4.10) can be re-written as

$$d_j^{L,(n)} = \sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{N_d} \breve{c}_i^{(n)} \frac{a_{ij}^2}{\lambda_{ij}^{(n)}} \left(\sum_{\substack{l=1 \\ a_{il} \neq 0}}^{N_p} \lambda_{il}^{(n)} \right). \tag{4.14}$$

Summations involving the constraint $a_{ij} \neq 0$ require knowledge of the projection geometry, and thereby each summation can be viewed as a type of forward or back projection.

The standard choice [2, 41]:

$$\bar{\lambda}_{ij} = a_{ij}, \;\; \bar{\lambda}_{kj} = |c_{kj}|, \tag{4.15}$$

leads to

$$\bar{d}_j^{L,(n)} = \sum_{i=1}^{N_d} \breve{c}_i^{(n)} a_{ij} \left(\sum_{l=1}^{N_p} a_{il} \right), \tag{4.16}$$

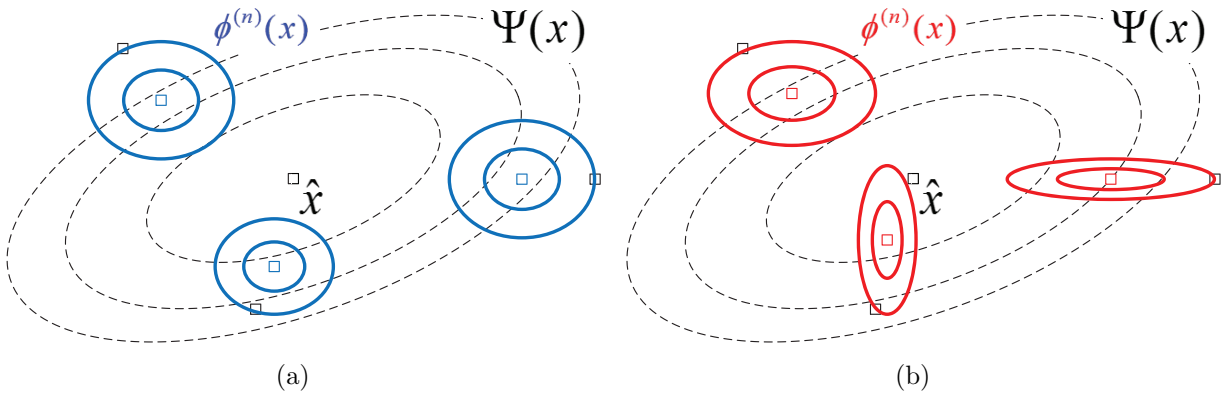
and

$$\bar{d}_j^{R,(n)} = \sum_{k=1}^{N_r} \ddot{\psi}_k(0)|c_{kj}| \left(\sum_{l=1}^{N_p} |c_{rl}| \right). \tag{4.17}$$

This choice is simple to implement, since the (available) standard forward and back projections can be used directly in (4.16). (Computing $\bar{d}_j^{R,(n)}$ in (4.17) is negligible compared with (4.16).) The standard SQS generates a sequence $\left\{x^{(n)}\right\}$ in (4.12) by defining the denominator as

$$\bar{d}_j^{(n)} \triangleq \bar{d}_j^{L,(n)} + \bar{d}_j^{R,(n)}. \tag{4.18}$$

However, we prefer choices for $\lambda_{ij}^{(n)}$ (and $\lambda_{kj}^{(n)}$) that provide fast convergence. Therefore, we first analyze the convergence rate of the SQS algorithm in terms of the choice of $\lambda_{ij}^{(n)}$ in Section 4.2.1. Then, Section 4.2.2 introduces acceleration by choosing better $\lambda_{ij}^{(n)}$ (and $\lambda_{kj}^{(n)}$) than the standard choice (4.15), which we name as a spatially nonuniform SQS (NU-SQS). Fig. 4.2 extends the illustration in Fig. 4.1 to present the intuition of the proposed NU-SQS methods that encourage larger updates for the voxels that need more updates for acceleration from standard SQS, with appropriately chosen $\lambda_{ij}^{(n)}$ (and $\lambda_{kj}^{(n)}$) based on convergence analysis discussed in next sextion.



(a) (b)

Figure 4.2: 2D illustration of SQS method: (a) Standard SQS methods using (4.16) construct a SQS surrogate $\phi^{(n)}$ with a similar shape regardless of the current location $x^{(n)}$ respect to the location of the minimizer $\hat{x}$. (b) Proposed nonuniform (NU) SQS methods are expected to construct a SQS surrogate that is adaptively adjusted based on the current $x^{(n)}$ respect to $\hat{x}$ to encourage larger updates for the voxels that need more updates.

We also provide two other variants of SQS in Section 4.3.

4.2 Spatially nonuniform SQS methods

We provide the convergence analysis of SQS algorithm, and propose NU approach that leads to faster convergence rate based on the analysis. The convergence rate of the sequence $\left\{x^{(n)}\right\}$ generated by the SQS iteration (4.12) depends on the denominator $D^{(n)} \triangleq \mathsf{diag}\left\{d_j^{(n)}\right\}$ (or $\lambda_{ij}^{(n)}$), and the main goal is to choose $\lambda_{ij}^{(n)}$ so that the sequence $\left\{x^{(n)}\right\}$ converges faster.

4.2.1 Convergence rate of SQS methods

The asymptotic convergence rate of a sequence $\left\{x^{(n)}\right\}$ that converges to $x^{(\infty)}$ is measured by the root-convergence factor defined as $R_1\left\{x^{(n)}\right\} \triangleq \limsup_{n\to\infty} \left|\left|x^{(n)} - x^{(\infty)}\right|\right|^{1/n}$ in [91, p. 288]. The root-convergence factor at $x^{(\infty)}$ for SQS algorithm is given as $R_1\left\{x^{(n)}\right\} = \rho\left(I - [D^{(\infty)}]^{-1}H^{(\infty)}\right)$ in [91, Linear Convergence Theorem, p. 301] and [39, Theorem 1], where the spectral radius $\rho(\cdot)$ of a square matrix is its largest absolute eigenvalue and $H^{(\infty)} \triangleq \nabla^2\Psi(x^{(\infty)})$, assuming that $D^{(n)}$ converges to $D^{(\infty)}$. For faster convergence, we want $R_1\left\{x^{(n)}\right\}$ and $\rho(\cdot)$ to be smaller. We can reduce the root-convergence factor based on[1] [39, Lemma 1], by using a smaller denominator $D^{(n)}$ subject to the majorization conditions in (2.15) or (4.8).

However, the asymptotic convergence rate does not help us design $D^{(n)}$ in the early iterations, so we consider another factor that relates to the convergence rate of SQS:

Lemma 1. *For a fixed denominator D (using the maximum curvature (4.5)), a sequence $\left\{x^{(n)}\right\}$ generated by an SQS algorithm (4.12) satisfies*

$$\Psi(x^{(n+1)}) - \Psi(x^{(\infty)}) \leq \frac{\left|\left|x^{(0)} - x^{(\infty)}\right|\right|_D^2}{2(n+1)}, \tag{4.19}$$

for any $n \geq 0$, if $\Psi(x)$ is convex.

Lemma 1 is a simple generalization of Theorem 3.1 in [10], which was shown for a surrogate with a scaled identity Hessian (using Lipschitz constant).[2] The inequality (4.19) shows that minimizing $\left|\left|x^{(0)} - x^{(\infty)}\right|\right|_D$ with respect to D will reduce the upper bound of $\Psi(x^{(n)}) - \Psi(x^{(\infty)})$, and thus accelerate convergence. (Since the upper bound is not tight, there should be a room for further acceleration by choosing better D, but we leave it as future work.)

[1] If $D_s^{-1} \succeq D_l^{-1} \succeq H^{(\infty)} \succeq 0$, then $\rho(I - D_s^{-1}H^{(\infty)}) \leq \rho(I - D_l^{-1}H^{(\infty)}) < 1$.

[2] Note that Lemma 1 can be further generalized for any positive definite matrix D, where the surrogate with such Hessian D satisfies the conditions (2.15) and (2.18). Here, we focus on a diagonal matrix D because the corresponding SQS algorithm leads to efficient implementation.

We want to adaptively design $D^{(n)}$ to accelerate convergence at the nth iteration. We can easily extend Lemma 1 to Corollary 1 by treating the current estimate $x^{(n)}$ as an initial image for the next SQS iteration:

Corollary 1. *A sequence* $\left\{x^{(n)}\right\}$ *generated by an SQS algorithm* (4.12) *satisfies*

$$\Psi(x^{(n+1)}) - \Psi(x^{(\infty)}) \leq \frac{\left|\left|x^{(n)} - x^{(\infty)}\right|\right|^2_{D^{(n)}}}{2} \tag{4.20}$$

for any $n \geq 0$, *if* $\Psi(x)$ *is convex.*

The inequality (4.20) motivates us to use $\left|x_j^{(n)} - x_j^{(\infty)}\right|$ when selecting $d_j^{(n)}$ (and $\lambda_{ij}^{(n)}$) to accelerate convergence at nth iteration. We discuss this further in Section 4.2.2.1. We fix $D^{(n)}$ after the n_{fix} number of iterations to ensure convergence of SQS iteration (4.12), based on [53]. In this case, $D^{(n)}$ must be generated by the maximum curvature (4.5) to guarantee the majorization condition (2.15) for subsequent iterations.

From (4.12) and (4.14), the step size $\Delta_j^{(n)}$ of the SQS iteration (4.12) has this relationship:

$$\Delta_j^{(n)} \triangleq x_j^{(n+1)} - x_j^{(n)} \propto \frac{1}{d_j^{(n)}}, \tag{4.21}$$

where smaller $d_j^{(n)}$ (and relatively larger $\lambda_{ij}^{(n)}$) values lead to larger steps. Therefore, we should encourage $d_j^{(n)}$ to be small ($\lambda_{ij}^{(n)}$ to be relatively large) to accelerate the SQS algorithm. However, we cannot reduce $d_j^{(n)}$ simultaneously for all voxels, due to the majorization conditions in (2.15) and (4.8). Lemma 1 (and Corollary 1) suggest intuitively that we should try to encourage larger steps $\Delta_j^{(n)}$ (smaller $d_j^{(n)}$) for the voxels that are far from the optimum to accelerate convergence.

4.2.2 NU-SQS methods

We design surrogates that satisfy condition (2.15) and provide faster convergence based on Section 4.2.1. We introduce the "*update-needed factors*" and propose a spatially nonuniform SQS (NU-SQS) algorithm.

4.2.2.1 Update-needed factors

Based on Corollary 1, knowing $\left|x_j^{(n)} - x_j^{(\infty)}\right|$ would be helpful for accelerating convergence at the nth iteration, but $x_j^{(\infty)}$ is unavailable in practice. Non-homogeneous coordinate descent (NH-CD) algorithm [109] used the difference between the current and previous iteration

instead:

$$u_j^{(n)} \triangleq \max\left\{\left|x_j^{(n)} - x_j^{(n-1)}\right|,\ \delta^{(n)}\right\}, \tag{4.22}$$

which we call the "*update-needed factors*" (originally named a voxel selection criterion (VSC) in [109]). Including the small positive values $\left\{\delta^{(n)}\right\}$ ensures all voxels to have at least a small amount of attention for updates. This $u_j^{(n)}$ accelerated the NH-CD algorithm by visiting voxels with large $u_j^{(n)}$ more frequently.

4.2.2.2 Design

For SQS, we propose to choose $\lambda_{ij}^{(n)}$ to be larger if the jth voxel is predicted to need more updates based on the "update-needed factors" (4.22) after the nth iteration. We select

$$\tilde{\lambda}_{ij}^{(n)} = a_{ij} u_j^{(n)}, \tag{4.23}$$

which is proportional to $u_j^{(n)}$ and satisfies the condition for $\lambda_{ij}^{(n)}$. This choice leads to the following NU-based denominator:

$$\tilde{d}_j^{L,(n)} = \frac{1}{u_j^{(n)}} \sum_{i=1}^{N_d} \breve{c}_i^{(n)} a_{ij} \left(\sum_{l=1}^{N_p} a_{il} u_l^{(n)} \right), \tag{4.24}$$

which leads to spatially nonuniform updates $\Delta_j^{(n)} \propto u_j^{(n)}$.

If it happened that

$$\left|x_j^{(n)} - x_j^{(\infty)}\right| \approx B \left|x_j^{(n)} - x_j^{(n-1)}\right| \text{ for all } j, \tag{4.25}$$

where B is a constant, then the NU denominator $\tilde{d}_j^{L,(n)}$ would minimize the upper bound of $\Psi(x^{(n+1)}) - \Psi(x^{(\infty)})$ in Corollary 1:

Lemma 2. *The proposed choice* $\tilde{d}_j^{L,(n)}$ *in* (4.24) *minimizes the following weighted sum of the denominators*

$$\sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 d_j^{L,(n)} \tag{4.26}$$

over all possible choices of the $d_j^{L,(n)}$ *in* (4.14).

Proof. In Appendix A. □

The proposed $\tilde{d}_j^{L,(n)}$ in (4.24) reduces to the standard choice $\bar{d}_j^{L,(n)}$ in (4.16) when $\left\{u_j^{(n)}\right\}$ is uniform. Similar to the standard choice $\bar{d}_j^{L,(n)}$, the proposed choice $\tilde{d}_j^{L,(n)}$ can be implemented

easily using standard forward and back projection. However, since $\tilde{d}_j^{L,(n)}$ depends on iteration (n), additional projections required for $\tilde{d}_j^{L,(n)}$ at every iteration would increase computation. We discuss ways to reduce this burden in Section 4.2.2.6.

Similar to the data-fit term, we derive the denominator of NU-SQS for the regularizer term to be:

$$\tilde{d}_j^{R,(n)} = \frac{1}{u_j^{(n)}} \sum_{k=1}^{N_r} \ddot{\psi}_k(0)|c_{kj}| \left(\sum_{l=1}^{N_p} |c_{rl}| u_l^{(n)} \right), \tag{4.27}$$

from the choice $\tilde{\lambda}_{kj}^{(n)} = |c_{kj}| u_j^{(n)}$ and the maximum curvature method in [2]. Alternatively, we may use Huber's optimal curvature [51, Lemma 8.3, p.184] replacing $\ddot{\psi}_k(0)$ in (4.27) by $\dot{\psi}_k\left([Cx^{(n)}]_k\right)/[Cx^{(n)}]_k$. The computation of (4.27) is much less than that of the data-fit term.

Defining the denominator in the SQS iteration (4.12) as

$$\tilde{d}_j^{(n)} \triangleq \tilde{d}_j^{L,(n)} + \tilde{d}_j^{R,(n)} \tag{4.28}$$

leads to the accelerated NU-SQS iteration, while the algorithm monotonically decreases $\Psi(x)$ and is provably convergent [53]. We can further accelerate NU-SQS by OS methods [2, 52] in Chapter III, while losing the guarantee of monotonicity. This algorithm, called OS algorithms based on a spatially nonuniform SQS (NU-OS-SQS), generates a sequence $\left\{x_j^{(n+\frac{m}{M})}\right\}$ using

$$x_j^{(n+\frac{m+1}{M})} = \left[x_j^{(n+\frac{m}{M})} - \frac{1}{\tilde{d}_j^{(n)}} \left(\gamma_j \frac{\partial}{\partial x_j} L_m\left(x^{(n+\frac{m}{M})}\right) + \frac{\partial}{\partial x_j} R\left(x^{(n+\frac{m}{M})}\right) \right) \right]_+, \tag{4.29}$$

where γ_j is computed from (3.6) (see Chapter III for more details of OS method).

4.2.2.3 Dynamic range adjustment of $u_j^{(n)}$

In reality, (4.25) will not hold, so (4.22) will be suboptimal. We could try to improve (4.22) by finding a function $f^{(n)}(\cdot) : [\delta^{(n)}, \infty) \to [\epsilon, 1]$ based on the following:

$$\underset{f^{(n)}(\cdot)}{\arg\min} \sum_{j=1}^{N_p} \left(f^{(n)}(u_j^{(n)}) - \frac{\left|x_j^{(n)} - x_j^{(\infty)}\right|}{\max_j \left|x_j^{(n)} - x_j^{(\infty)}\right|} \right)^2, \tag{4.30}$$

where ϵ is a small positive value. Then we could use $f^{(n)}(u_j^{(n)})$ as (better) update-needed factors. However, solving (4.30) is intractable, so we searched empirically for good candidates for a function $f^{(n)}(\cdot)$.

Intuitively, if the dynamic range of the update-needed factors $u_j^{(n)}$ in (4.22) is too wide, then there will be too much focus on the voxels with relatively large $u_j^{(n)}$, slowing the overall

convergence rate. On the other hand, a narrow dynamic range of $u_j^{(n)}$ will provide no speed-up, since the algorithm will distribute its efforts uniformly. Therefore, adjusting the dynamic range of the update-needed factors is important to achieve fast convergence. This intuition corresponds to how the NH-CD approach balanced between homogeneous update orders and nonhomogeneous update orders [109].

To adjust the dynamic range and distribution of $u_j^{(n)}$, we first construct their empirical cumulative density function:

$$F_{\text{cdf}}^{(n)}(u) \triangleq \frac{1}{N_p} \sum_{j=1}^{N_p} I_{\left\{u_j^{(n)} \leq u\right\}} \tag{4.31}$$

to somewhat normalize their distribution, where $I_B = 1$ if B is true or 0 otherwise. Then we map the values of $F_{\text{cdf}}^{(n)}(u)$ by a nondecreasing function $g(\cdot) : [0,\, 1] \to [\epsilon,\, 1]$ as follows

$$\tilde{u}_j^{(n)} \triangleq f^{(n)}(u_j^{(n)}) = g\left(F_{\text{cdf}}^{(n)}(u_j^{(n)})\right), \tag{4.32}$$

which controls the dynamic range and distribution of $\left\{\tilde{u}_j^{(n)}\right\}_{j=1}^{N_p}$, and we enforce positivity in $g(\cdot)$ to ensure that the new adjusted parameter $\tilde{\lambda}_{ij}^{(n)} = a_{ij}\tilde{u}_j^{(n)}$ is positive if a_{ij} is positive. (We set $\delta^{(n)}$ in (4.22) to zero here, since a positive parameter ϵ ensures the positivity of $\tilde{\lambda}_{ij}^{(n)}$ if a_{ij} is positive.) The transformation (4.32) from $u_j^{(n)}$ to $\tilde{u}_j^{(n)}$ is called dynamic range adjustment (DRA), and two examples of such $\tilde{u}_j^{(n)}$ are presented in Fig. 4.3. Then we use $\tilde{u}_j^{(n)}$ instead of $u_j^{(n)}$ in (4.23).

Here, we focus on the following function for adjusting the dynamic range and distribution:

$$g(v) \triangleq \max\left\{v^t,\, \epsilon\right\} \tag{4.33}$$

where t is a nonnegative real number that controls the distribution of $\tilde{u}_j^{(n)}$ and ϵ is a small positive value that controls the maximum dynamic range of $\tilde{u}_j^{(n)}$. The function reduces to the ordinary SQS choice in (4.15) when $t = 0$. The choice of $g(\cdot)$, particularly the parameters t and ϵ here, may influence the convergence rate of NU-SQS for different data sets, but we show that certain values for t and ϵ consistently provide fast convergence for various data sets.

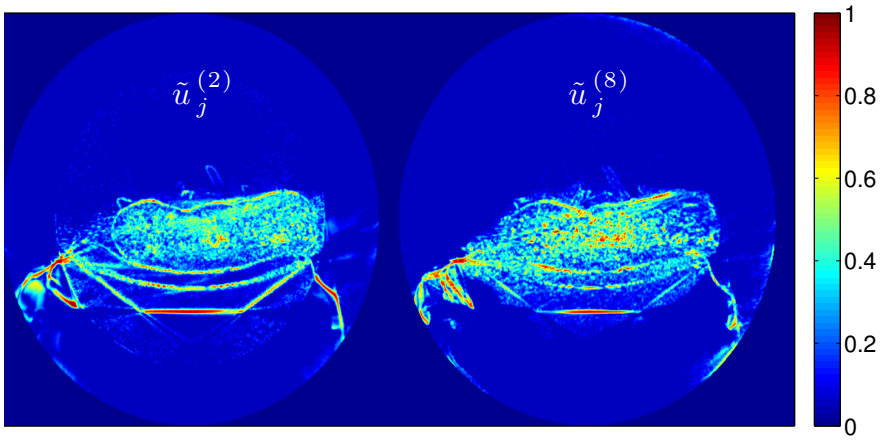


Figure 4.3: Shoulder region scan: $\tilde{u}_j^{(2)}$ and $\tilde{u}_j^{(8)}$ after dynamic range adjustment (DRA) for NU-OS-SQS(82 subsets), with the choice $g(v) = \max\{v^{10}, 0.05\}$. NU-OS-SQS updates the voxels with large $\tilde{u}_j^{(n)}$ more, whereas ordinary OS-SQS updates all voxels equivalently.

4.2.2.4 Related work

In addition to the standard choice (4.15), the choice

$$\lambda_{ij}^{(n)} = a_{ij} \max\left\{x_j^{(n)},\ \delta\right\}, \tag{4.34}$$

with a small nonnegative δ, has been used in emission tomography problems [24, 25] and in transmission tomography problems [41, 66]. This choice is proportional to $x_j^{(n)}$, and thereby provides a relationship $\Delta_j^{(n)} \propto x_j^{(n)}$. This classical choice (4.34) can be also viewed as another NU-SQS algorithm based on "intensity". However, intensity is not a good predictor of which voxels need more update, so (4.34) does not provide fast convergence based on the analysis in Section 4.2.1.

4.2.2.5 Initialization of $u_j^{(0)}$

Unfortunately, $u_j^{(n)}$ in (4.22) is available only for $n \geq 1$, *i.e.*, after updating all voxels once. To define the initial update factors $u_j^{(0)}$, we apply edge and intensity detectors to an initial filtered back-projection (FBP) image. This is reasonable since the initial FBP image is a good low-frequency estimate, so the difference between initial and final image will usually be larger near edges. We investigated one particular linear combination of edge and intensity information from an initial image. We used the 2D Sobel operator to approximate the magnitude of the gradient of the image within each transaxial plane. Then we scaled both the magnitude of the approximated gradient and the intensity of the initial image to have a same maximum value, and computed a linear combination of two arrays with a ratio

$2:1$ for the initial update-needed factor $u_j^{(0)}$, followed by DRA method. We have tried other linear combinations with different ratios, but the ratio $2:1$ provided the fastest convergence rate in our experiments.

4.2.2.6 Implementation

The dependence of $\lambda_{ij}^{(n)}$ on iteration (n) increases computation, but we found two practical ways to reduce the burden. First, we found that it suffices to update $\tilde{u}_j^{(n)}$ (and $\tilde{d}_j^{(n)}$) every $n_{\text{loop}} > 1$ iterations instead of every iteration. This is reasonable since the update-needed factors usually change slowly with iteration. In this case, we must generate a surrogate with the maximum curvature (4.5) to guarantee the majorization condition (2.15) for all iterations. Second, we compute the NU-based denominator (4.24) simultaneously with the data-fit gradient in (4.12). In 3D CT, we use forward and back projectors that compute elements of the system matrix A on the fly, and those elements are used for the gradient $\nabla L(x)$ in (4.12). For efficiency, we reuse those computed elements of A for the NU-based denominator (4.24). We implemented this using modified separable footprint projector subroutines [70] that take two inputs and project (or back-project) both. This approach required only 29% more computation time than a single forward projection rather than doubling the time (see Table 4.1). Combining this approach with $n_{\text{loop}} = 3$ yields a NU-SQS algorithm that required only 11% more computation time per iteration than standard SQS, but converges faster.

	SQS	OS-SQS(82)	NU-OS-SQS(82)		
n_{loop}	·	·	1	3	5
1 Iter. [sec]	82	125	161	139	133

Table 4.1: Run time of one iteration of NU-OS-SQS(82 subsets) for different choice of n_{loop} for GE performance phantom.

Computing $\tilde{u}_j^{(n)}$ and the corresponding NU-based denominator requires one iteration each. In the proposed algorithm, we computed $\tilde{u}_j^{(n)}$ during one iteration, and then computed the NU-based denominator (4.24) during the next iteration combined with the gradient computation $\nabla L(x)$. Then we used the denominator for n_{loop} iterations and then compute $\tilde{u}_j^{(n)}$ again to loop the process. The outline of the proposed NU-OS-SQS algorithm is described in the following Table 4.2.

Set M, n_{end}, n_{loop} and initialize x by an FBP image.
Generate u_j from an FBP image by edge and intensity detectors.
Compute the maximum curvature $\breve{c}_i = \max\left\{\ddot{h}_i(0),\ \eta\right\}$.
$d_j^L = 0$, $\gamma_j = 0$, $x_{j,\text{ref}} = x_j$, and the final image $\bar{x}_j = 0$.

$$\tilde{d}_j^R = \frac{1}{u_j}\sum_{k=1}^{N_r} \ddot{\psi}_k(0)|c_{kj}| \left(\sum_{l=1}^{N_p} |c_{rl}|u_l\right) \tag{4.35}$$

$n = 0$
for $m = 0, 1, \ldots, M-1$

$$d_{j,\text{sub}}^L = \frac{1}{u_j}\sum_{i\in S_m} \breve{c}_i a_{ij} \left(\sum_{l=1}^{N_p} a_{il}u_l\right) \tag{4.36}$$

$$d_j^L \mathrel{+}= d_{j,\text{sub}}^L \text{ and } \gamma_j \mathrel{+}= I_{\left\{d_{j,\text{sub}}^L > 0\right\}}$$

end

Table 4.2: Outline of the proposed NU-OS-SQS algorithm (cont'd).

for $n = 1, 2, \ldots, n_{\text{end}} - 1$

if $n \bmod n_{\text{loop}} = 1$ and $n \leq n_{\text{fix}}$

$$\tilde{d}_j^L = d_j^L, \text{ and compute } \tilde{d}_j^R \text{ by (4.35)}$$

elseif $n \bmod n_{\text{loop}} = n_{\text{loop}} - 1$ and $n \leq n_{\text{fix}} - 2$

$$x_{j,\text{ref}} = x_j$$

elseif $n \bmod n_{\text{loop}} = 0$ and $n \leq n_{\text{fix}} - 1$

$$d_j^L = 0, \text{ and } u_j = g(F_{\text{cdf}}(x_j - x_{j,\text{ref}}))$$

end

for $m = 0, 1, \ldots, M - 1$

$$x_{j,\text{prev}} = x_j$$

if $n \bmod n_{\text{loop}} \neq 0$ or $n \geq n_{\text{fix}}$

$$g_{j,\text{sub}}^L = \frac{\partial}{\partial x_j} L_m(x_{\text{prev}}) \tag{4.37}$$

else

$$\text{compute both } d_{j,\text{sub}}^L \text{ by (4.36) and } g_{j,\text{sub}}^L \text{ by (4.37)}$$

$$\text{simultaneously using two-input projection function, and}$$

$$d_j^L \mathrel{+}= d_{j,\text{sub}}^L$$

end

$$x_j = \left[x_{j,\text{prep}} - \frac{\gamma_j g_{j,\text{sub}}^L + \frac{\partial}{\partial x_j} R(x_{\text{Trev}})}{\tilde{d}_j^L + \tilde{d}_j^R} \right]_+ \tag{4.38}$$

if $n = n_{\text{end}} - 1$

$$\bar{x}_j = \frac{m}{m+1} \bar{x}_j + \frac{1}{m+1} x_j$$

end

end

end

Table 4.2: Outline of the proposed NU-OS-SQS algorithm.

4.2.3 Results

We investigated the proposed NU-OS-SQS algorithm for PWLS image reconstruction (2.10) with a nonnegativity constraint. The PWLS cost function is strictly convex and has a unique global minimizer [27]. We implemented the NU-OS-SQS algorithm in C and executed it on a Mac with two 2.26GHz quad-core Intel Xeon processors and a 16GB RAM. We used 16 threads, and projection views were grouped and assigned to each thread. We used the approaches in Chapter III that are developed for OS method in 3D helical geometry, the scaling factor for helical geometry in Section 3.2 and the averaging technique in Section 3.3.

Four 3D helical CT data sets are used in this section to compare the proposed NU-OS-SQS algorithm to the ordinary OS-SQS algorithm, and we used the GE performance phantom (GEPP) to measure the resolution. We used two other clinical data sets to investigate the performance of NU approach. We investigated tuning the DRA function $g(\cdot)$ in (4.33) to provide fast convergence rate for various data sets. We also provide results from a simulation data set.

We chose the parameters of the cost function $\Psi(x)$ in (2.8) to provide a good image. We defined an edge-preserving potential function $\psi_k([Cx]_k) = \beta_k \psi([Cx]_k)$ by the function $\psi(t)$ in (2.13) (with $a = 0.0558$, $b = 1.6395$, and $\delta = 10$), and the spatial weighting β_k [43] that provides resolution properties emulating the GE product "Veo". We used $M = 82$ subsets for the OS algorithms, assigning 12 out of 984 projection views per rotation to each subset. We used the maximum curvature (4.5) for generating the denominator of surrogate function of the cost function $\Psi(x)$, and focused on $n_{\text{loop}} = 3$ which balances the convergence rate and run time, based on Table 4.1.

In Section 4.2.1, we recommended fixing the denominator $\tilde{d}_j^{(n)}$ (generated by the maximum curvature (4.5)) after n_{fix} iterations in NU-SQS algorithm to guarantee convergence. This condition is less important theoretically when we accelerate the NU-SQS algorithm with OS methods that break the convergence property. However, we still recommend fixing $\tilde{d}_j^{(n)}$ after n_{fix} iterations (before approaching the limit-cycle) in the NU-OS-SQS algorithm, because we observed some instability from updating $\tilde{d}_j^{(n)}$ (and $\tilde{u}_j^{(n)}$) every n_{loop} iterations near the limit-cycle in our experiments. We selected $n_{\text{fix}} = 7$ for GEPP, but we did not use n_{fix} for other two cases because the algorithm did not reach a limit-cycle within $n_{\text{end}} = 20$ iterations, and we leave optimizing n_{fix} as a future work.

In Section 3.2, we stabilized the OS-SQS algorithm outside ROI in helical geometry by using the factor γ_j in (3.6). However, we experienced some instability outside ROI in NU-OS-SQS methods even with (3.6), because a small NU denominator $\tilde{d}_j^{(n)}$ outside ROI is more likely to lead to instability than for voxels within the ROI due to the incomplete sampling outside ROI. Therefore, we prevent the denominator $\tilde{d}_j^{(n)}$ outside ROI from being

very small. We empirically modified the DRA function in Section 4.2.2.3, and used it for our experiments, improving stability outside ROI. We first modified the function (4.31) as follows:

$$F_{\text{cdf}}^{(n)}(u) \triangleq \frac{1}{N_p} \sum_{j=1}^{N_p} I_{\left\{\gamma_j u_j^{(n)} \leq u\right\}} \tag{4.39}$$

for γ_j in (3.6), since the value of $u_j^{(n)}$ in (4.22) outside ROI was found to be relatively large due to the incomplete sampling. We further modified (4.32) and (4.33) to prevent $\tilde{d}_j^{(n)}$ from becoming very small outside ROI as follows with $g(v;\, \alpha) \triangleq \max\{\alpha v^t,\, \epsilon\}$:

$$\tilde{u}_j^{(n)} \triangleq \begin{cases} g\left(F_{\text{cdf}}^{(n)}(\gamma_j u_j^{(n)});\, 1\right), & \text{if } j\text{th voxel within ROI,} \\ g\left(F_{\text{cdf}}^{(n)}(\gamma_j u_j^{(n)});\, 0.5\right), & \text{otherwise.} \end{cases} \tag{4.40}$$

4.2.3.1 GE performance phantom

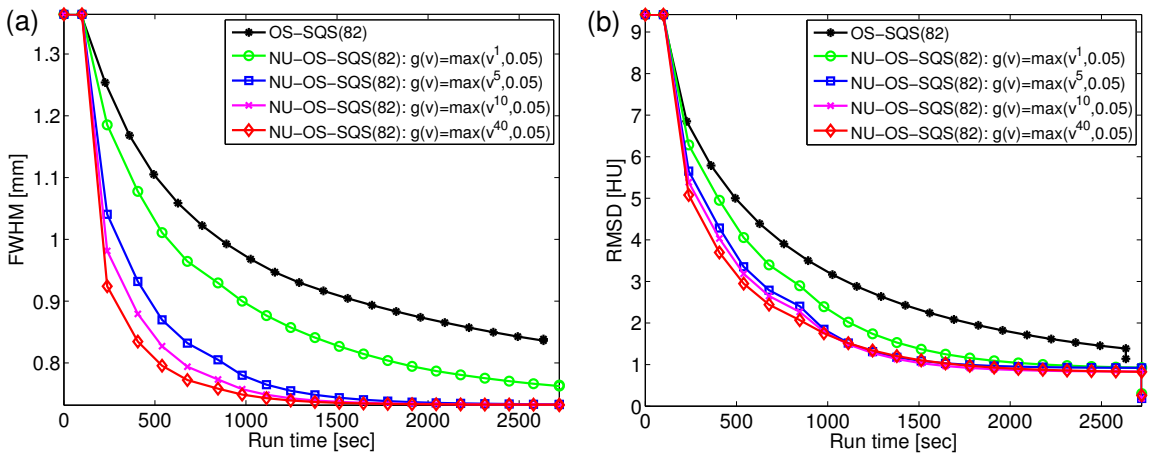
We reconstructed $512 \times 512 \times 47$ images of the GEPP from a $888 \times 64 \times 3071$ sinogram (the number of detector columns $\times$ detector rows $\times$ projection views) with pitch 0.5. We evaluated the full width at half maximum (FWHM) of a tungsten wire (see Fig. 3.2). Fig. 4.4(a) shows the resolution versus run time and confirms that nonuniform (NU) approach accelerates the SQS algorithm. This dramatic speed-up in FWHM is promising since SQS-type algorithms are known to have slow convergence rate of high frequency components [96]. We also evaluated the convergence rate by computing RMSD (3.8) between current and converged[3] image versus run time, within ROI.

Figs. 4.4(a) and 4.4(b) illustrate that increasing t in $g(\cdot)$ in (4.33) accelerates the convergence of "update-needed" region, particularly the wire and edges in GEPP. However, highly focusing the updates on few voxels will not help speed up the overall convergence for all objects. Therefore, we further investigate the choice of $g(\cdot)$ using various patient CT scans.

The RMSD plots[4] of NU-OS-SQS in Fig. 4.4(b) reached a limit-cycle after 1500 sec that did not approach zero. Averaging the sub-iterations at the final iteration that is suggested in Section 3.3 improved the final image with small computation cost, yielding the drop in RMSD at the last 20th iteration in Fig. 4.4(b). The reduced noise was measurable in the reconstructed image as seen in Table 3.1.

[3] We ran 100 iterations of OS-SQS algorithm with 41 subsets, followed by each 100 iterations of OS-SQS algorithm with 4 subsets, and 2000 iterations of (convergent) SQS. We subsequently performed 100 iterations of (convergent) NH-ABCD-SQS [57] to generate (almost) converged images $x^{(\infty)}$.

[4] We also provide the plots of the cost function for GEPP and shoulder region scan in Section 4.2.3.4.



(a) (b)

Figure 4.4: GE performance phantom: plots of (a) FWHM and (b) RMSD as a function of run time for different choice of DRA parameters t for $\epsilon = 0.05$. The plot markers show each iteration. There are no changes during first iterations, since we consider precomputing the denominator using one forward and back projections as one iteration.

4.2.3.2 Shoulder region scan

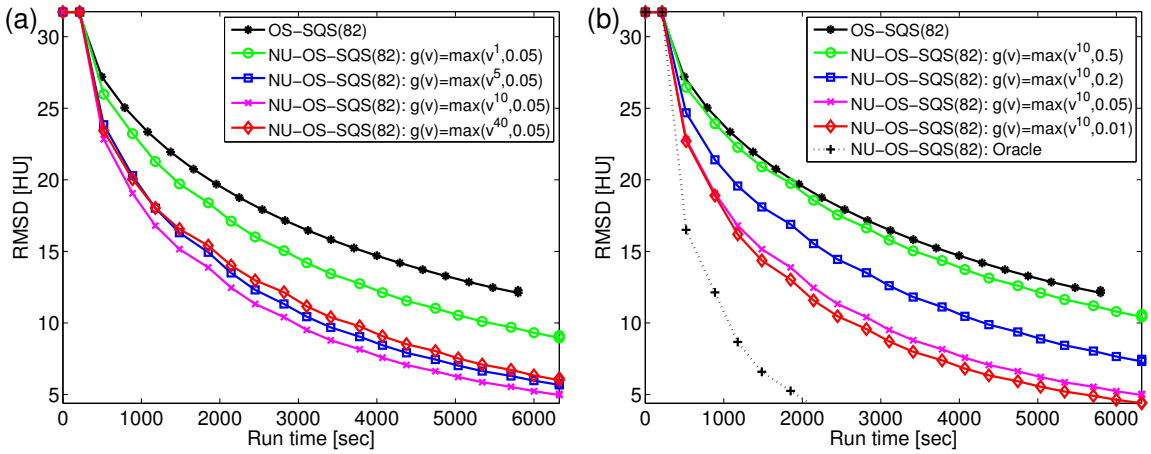
In this experiment, we reconstructed a $512 \times 512 \times 109$ image from a shoulder region scan $888 \times 32 \times 7146$ sinogram with pitch 0.5. Figs. 4.5(a) and 4.5(b) show that the nonuniform approach accelerates convergence, depending on the choice of parameters in $g(\cdot)$. We investigated the relationship between the convergence rate and the DRA function $g(\cdot)$ by tuning both the parameters t and ϵ in (4.33). Fig. 4.5(a) shows that increasing t to 10 accelerated convergence, but larger t values did not help as the choice of $t = 40$ was slower than $t = 10$. In Fig. 4.5(b), decreasing ϵ to 0.01 accelerated the algorithm in this shoulder region scan, but not for the data set in Section 4.2.3.3, so $\epsilon = 0.05$ appears to be a reasonable choice overall.

We averaged the sub-iterations at the last iteration, but Figs. 4.5(a) and 4.5(b) did not show a drop at the final iteration (which appeared in Fig. 4.4(b)), because the algorithm had not yet reached a limit-cycle. Even though averaging technique in Section 3.3 did not noticeably decrease the RMSD, the reconstructed image had measurable noise reduction in regions that already reached a limit-cycle like uniform regions. (Results not shown.)

In Fig. 4.6(a), we illustrate that statistical image reconstruction can reduce noise and preserve image features compared to analytical FBP reconstruction. The reconstructed images of (NU-)OS-SQS show that NU approach helps OS-SQS to approach the converged image faster than the ordinary method. After the same computation (95 min.), the reconstructed

image of OS-SQS still contains streaks from the initial FBP image, while NU-OS-SQS has reduced the streaks. This is apparent in the difference images between the reconstructed and converged images in Fig. 4.6(b).

By analyzing NU-OS-SQS in two CT scans, we observed that the parameters $t = 10$ and $\epsilon = 0.05$ consistently accelerated the algorithm by about a factor of more than two.[5] (The choice $\epsilon = 0.01$ was too aggressive in our other experiments.) We also have observed more than two-fold accelerations in other experiments. (Results not shown.) Fig. 4.5(b) shows the RMSD plot using the (practically unavailable) oracle update-needed factor $\hat{u}_j^{(n)} \triangleq |x_j^{(n)} - x_j^{(\infty)}|$ instead of our heuristic choice $\tilde{u}_j^{(n)}$. This result suggests that additional optimization of the DRA method and initialization of $\tilde{u}_j^{(0)}$ could further speed up the NU algorithm in future work.



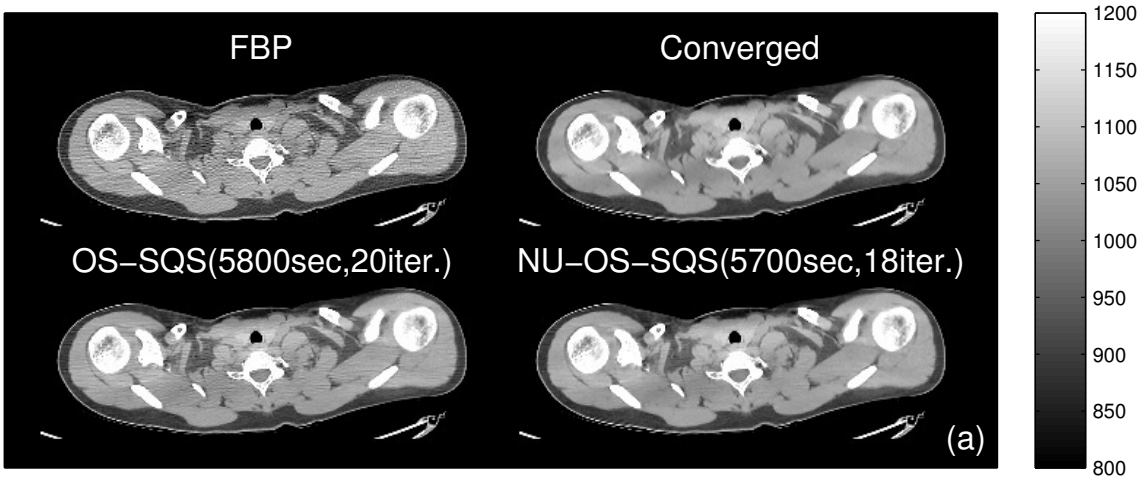
(a) (b)

Figure 4.5: Shoulder region scan: plot of RMSD versus run time for different choice of parameters (a) t and (b) ϵ in $g(v) = \max\{v^t, \epsilon\}$.

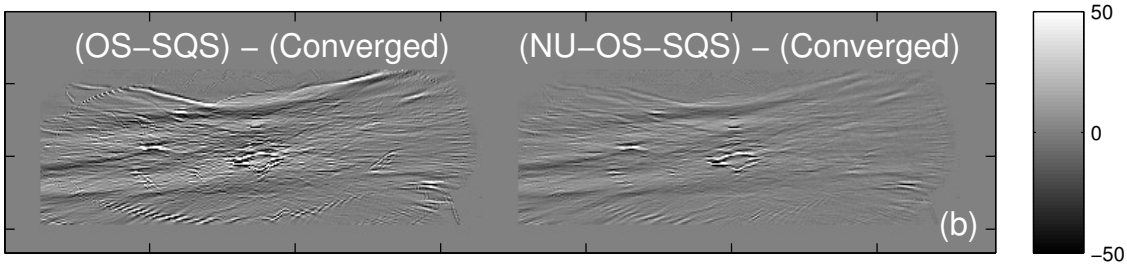
4.2.3.3 Truncated abdomen scan

We also reconstructed a $390 \times 390 \times 239$ image from a $888 \times 64 \times 3516$ sinogram with pitch 1.0. This scan contains transaxial truncation and the initial FBP image has truncation artifacts [108] that can be reduced by iterative reconstruction. The choice of $u_j^{(0)}$ described in Section 4.2.2.5 did not consider truncation effects, and we found that NU-OS-SQS did not reduce such artifacts faster than standard OS-SQS. (The large patient size may also have

[5] We used the run time and RMSD of standard OS-SQS after 20 iterations (without averaging) as a reference to compare with the NU-OS-SQS for each data set. Then we compared the run time of NU-OS-SQS that is required for achieving the reference RMSD with the reference run time, and confirmed that NU provided more than two-fold accelerations in two CT scans.



(a)



(b)

Figure 4.6: Shoulder region scan: (a) Center slice of initial FBP, converged image and reconstructed image by OS-SQS(82) and NU-OS-SQS(82)-$g(v) = \max\{v^{10}, 0.05\}$ after about 95 min. (b) Difference between the reconstructed and converged images are additionally shown to illustrate the acceleration of NU approach. (Images are cropped for better visualization.)

reduced the possible speed-up by the NU method, compared to the previous two scans.) Therefore, we investigated an alternative NU method that can reduce truncation artifacts faster than standard algorithm.

We designed a modified NU method using a few (m_{sub}) sub-iterations of standard OS-SQS to generate the initial update-needed factor $u_j^{(0)}$, which may also be a reasonable approach for other scans. We perform initial sub-iterations $x_{\text{sub}}^{(m/M)}$ in (4.29) efficiently using two-input projectors (in Section 4.2.2.6) and replacing the all-view denominator $\tilde{d}_j^{L,(n)}$ in (4.24) by a standard subset-based denominator [4]:

$$\tilde{d}_{j,\text{sub}}^{L_m,\left(\frac{m}{M}\right)} \triangleq \gamma \sum_{i \in S_m} \breve{c}_i^{\left(\frac{m}{M}\right)} a_{ij} \left(\sum_{l=1}^{N_p} a_{il} \right), \tag{4.41}$$

where S_m consists of projection views in mth subset. The scaling factor γ_j in (3.6) is

unavailable at this point, so we use $\gamma = M$ instead. After m_{sub} sub-iterations, we compute the following initial update-needed factors:

$$\tilde{u}_j^{(0)} \triangleq f_{\text{sub}}^{\left(\frac{m_{\text{sub}}}{M}\right)} \left(\left| x_{j,\text{sub}}^{\left(\frac{m_{\text{sub}}}{M}\right)} - x_j^{(0)} \right| \right), \tag{4.42}$$

where $f_{\text{sub}}^{\left(\frac{m_{\text{sub}}}{M}\right)}(\cdot)$ is a DRA function in (4.32), and we use these to compute the NU denominators $\tilde{d}_j^{L,(0)}$ and $\tilde{d}_j^{R,(0)}$ that we use for first n_{loop} outer iterations.

Fig. 4.7(a) shows that statistical image reconstruction provides better image quality than FBP reconstruction. Fig. 4.7(b) illustrates that this NUsub-OS-SQS approach reduces the truncation artifacts faster than the standard OS-SQS and NU-OS-SQS. Although standard OS-SQS reduces noise faster than other two algorithms in Fig. 4.7(b), both NU-OS-SQS and NUsub-OS-SQS show better convergence near the spine, the boundary of patient, and other internal structures than OS-SQS at the same computation time (90 min.).

Next, two sections are from the supplementary material of [62] that supports the previous experimental results.

4.2.3.4 Cost function plots

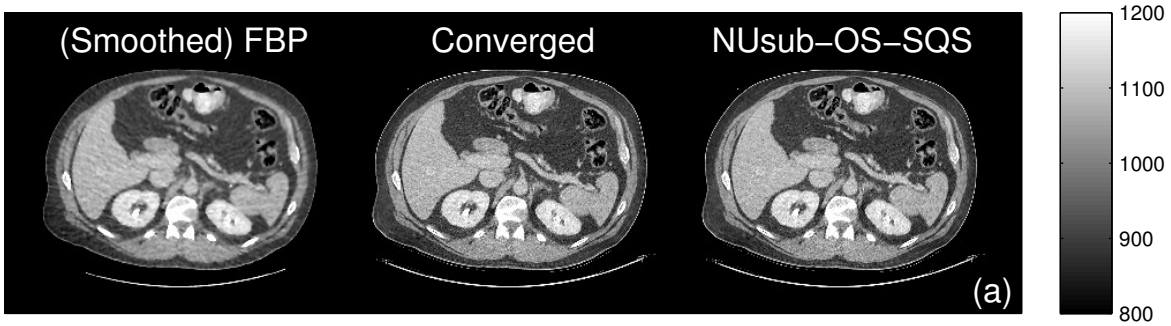
This section provides cost function plots for GEPP and shoulder region scan in Sections 4.2.3.1 and 4.2.3.2.

Previously, we computed RMSD within the ROI to evaluate the convergence rate of the proposed algorithm. Another way to assess the convergence rate is computing the cost function $\Psi(x)$ in (2.8) at each iteration. We used the following metric:

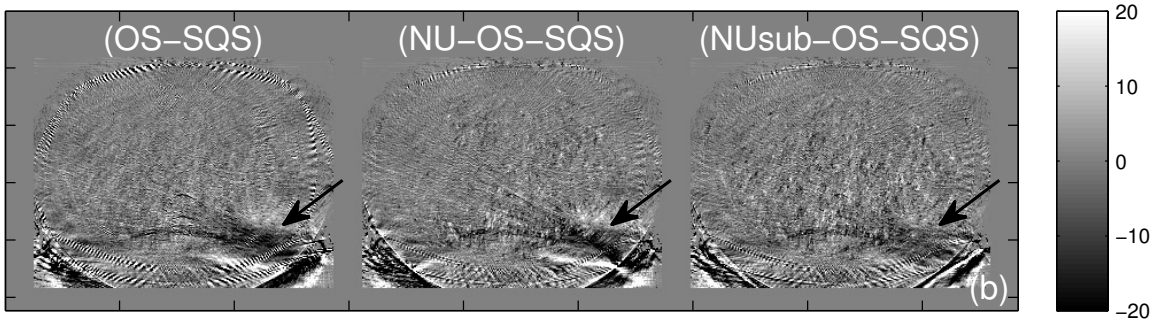
$$\xi^{(n)} = 20 \log_{10} \left(\frac{\Psi(x^{(n)}) - \Psi(x^{(\infty)})}{\Psi(x^{(\infty)})} \right) \text{ [dB]} \tag{4.43}$$

to better visualize how the cost function decreases each iteration. We used double precision and triple **for** loops when accumulating $\Psi(x^{(n)})$ to ensure high accuracy.

Fig. 4.8 shows plots of $\xi^{(n)}$ for the choices of parameters used in Figs. 4.4 and 4.5 for two real 3D scans; GEPP and shoulder region scan. Fig. 4.8(a) shows that for the GEPP case, the NU-OS-SQS methods decreased the cost function at about the same rate than the ordinary OS method, or even perhaps slightly slower. In contrast, when we plotted RMSD distance to the converged image within the ROI in Fig. 4.4, NU-OS-SQS converged significantly faster. The reason for this different behavior is that the cost function plot considers all voxels, even those outside the ROI which are not of interest clinically. It is known that OS methods are not guaranteed to converge and apparently the non-ROI voxels are either not converging



(a)



(b)

Figure 4.7: Truncated abdomen scan: (a) Center slice of FBP, converged image, and reconstructed image by NUsub-OS-SQS(82)-$g(v) = \max\{v^{10}, 0.05\}$ using $\tilde{u}_j^{(0)}$ in (4.42) generated from sub-iterations. (b) Difference between the reconstructed and converged images, where images are reconstructed by OS-SQS(82) after 5400sec (20iter.), NU-OS-SQS(82) after 5230sec (18iter.) using $\tilde{u}_j^{(0)}$ extracted from FBP based on Section 4.2.2.5, and NUsub-OS-SQS(82) after 5220sec (17iter.) using $\tilde{u}_j^{(0)}$ in (4.42). The (black) arrows indicate truncation artifacts. Images are cropped for better visualization.

or perhaps approaching a larger limit-cycle, presumably due to the poor sampling in the padded slices outside the ROI, even with the stabilizing methods outside ROI described in Section 3.2. Therefore, cost function plots may not provide practical measures of convergence rate for OS methods, particularly with acceleration. Future research on trying to further stabilize the NU-OS-SQS algorithm outside the ROI also may be helpful.

The final drops at the right in Fig. 4.8(a) show that averaging sub-iterations at the last iteration, as described in Section 3.3, can compensate for the limit-cycle, particularly outside the ROI.

Unlike Fig. 4.8(a), the plots in Fig. 4.8(b) and 4.8(c) of shoulder region scan look similar to the plots of RMSD within ROI in Fig. 4.5. The scan geometry of each data set might explain these behavior of cost function in Fig. 4.8, where the shoulder region scan is a helical scan with pitch 1.0 and 7 helical turns and thus the corresponding image space has relatively

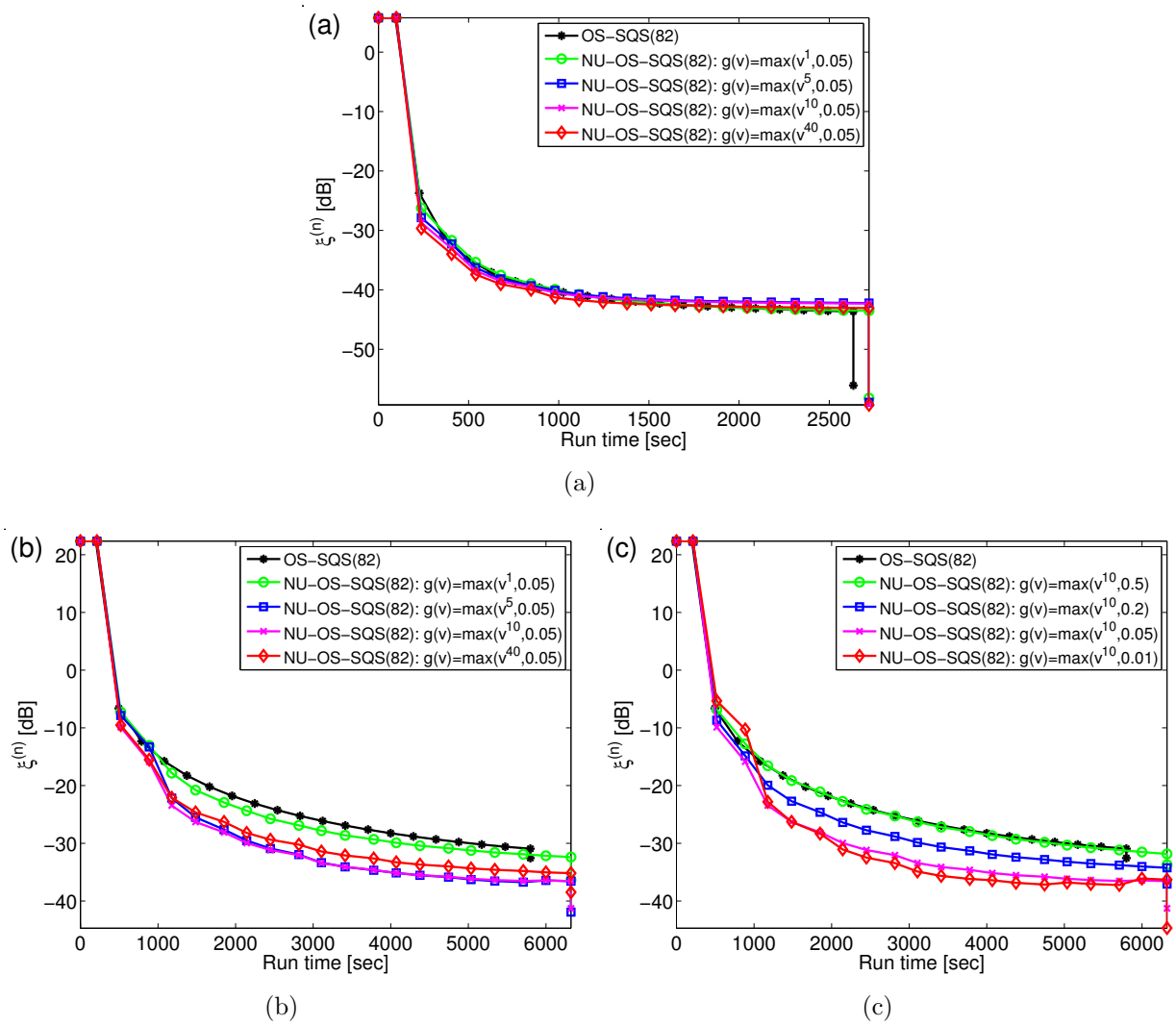


Figure 4.8: Plots of $\xi^{(n)}$ in (4.43) as a function of run time for different choice of DRA parameters for (a) GE performance phantom and (b-c) shoulder region scan.

few voxels outside the ROI, compared with GEPP data that is acquired by a helical scan with pitch 0.5 and 3 helical turns. Therefore, we can expect the cost function of shoulder region scan to be less affected by instability outside the ROI. Slower convergence of NU-OS-SQS algorithm at early iterations in Fig. 4.8(c) means that some choices of initial update-needed factor $\tilde{u}_j^{(0)}$ were not good enough for voxels outside the ROI. The effect of averaging at the last iterations is apparent in Fig. 4.8(b) and 4.8(c), because the instability outside the ROI is suppressed by the averaging.

4.2.3.5 Simulation data

This section provides a simulation study of a helical scan of the XCAT phantom [98] for the reproducability of the results. We first acquired a $1024 \times 1024 \times 154$ XCAT phantom for 500 [mm] transaxial field-of-view (FOV) at 70 [keV], where $\Delta_x = \Delta_y = 0.4883$ [mm] and $\Delta_z = 0.6250$ [mm]. (See Fig. 4.9.)

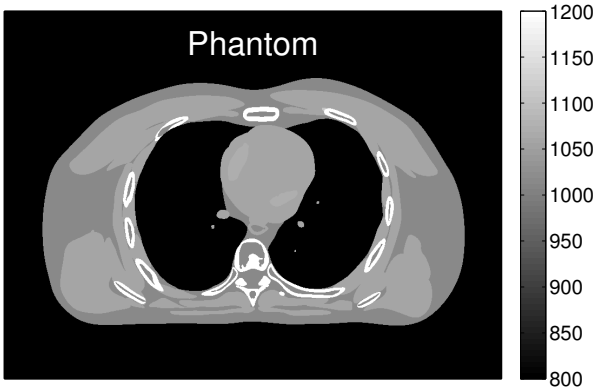


Figure 4.9: Simulated XCAT phantom: a center slice of $1024 \times 1024 \times 154$ XCAT phantom. (Images are cropped for better visualization.)

We simulated a helical scan using the blank scan factor $b_i = 10^6$ and the mean number of background events $r_i = 0$ with Poisson noise. The sinogram data is in $888 \times 64 \times 2934$ (the number of detector columns×detector rows×projection views) space with pitch 1.0. Then, we reconstructed a $512 \times 512 \times 154$ image where $\Delta_x = \Delta_y = 0.9766$ [mm] and $\Delta_z = 0.6250$ [mm] using the proposed NU-OS-SQS algorithm.

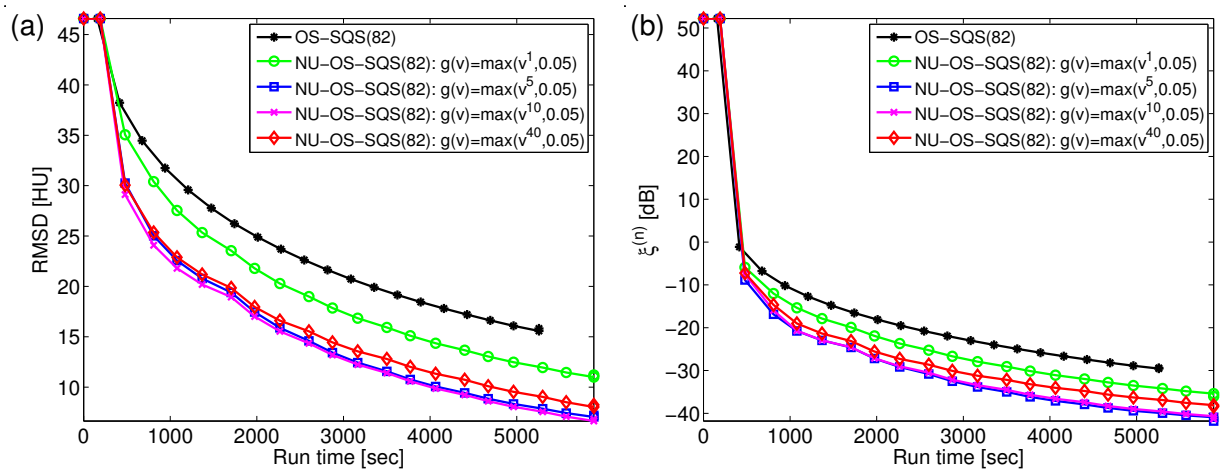
We solve a PWLS function with a potential function $\psi_k(t) \triangleq \beta_k \psi(t)$ in (2.13) using a spatial weighting parameter:

$$\beta_k \triangleq 50 \cdot \prod_{\substack{j=1 \\ c_{kj} \neq 0}}^{N_p} \max\left\{\kappa_j,\ 0.01\ \kappa_{\max}\right\} \tag{4.44}$$

that provides uniform resolution properties [43], where

$$\kappa_j \triangleq \sqrt{\frac{\sum_{i=1}^{N_d} a_{ij} w_i}{\sum_{i=1}^{N_d} a_{ij}}} \tag{4.45}$$

and the value of $\kappa_{\max} \triangleq \max_j \kappa_j$ is used in (4.44) to avoid under-regularizing some voxels with very small κ_j. Fig. 4.10 illustrates both RMSD within ROI and $\xi^{(n)}$ versus computation time.



(a) (b)

Figure 4.10: Simulated XCAT phantom: plots of (a) RMSD and (b) $\xi^{(n)}$ versus run time for different choice of parameters t for $\epsilon = 0.05$ in $g(v) = \max\{v^t, \epsilon\}$.

In Fig. 4.10(a), we evaluated the convergence rate using RMSD within ROI between current and converged image, where the converged image was generated by many iterations of a (convergent) SQS. We used parameters of DRA function that are used in Figs. 4.4 and 4.5, and we observed similar trends. We also illustrate the plot of $\xi^{(n)}$ versus run time in Fig. 4.10(b), which looks very similar to Fig. 4.10(a). This is because we regularized relatively more than two other experiments in this simulation experiment, and thus instability outside the ROI that can be caused by NU-OS-SQS methods is not apparent here.

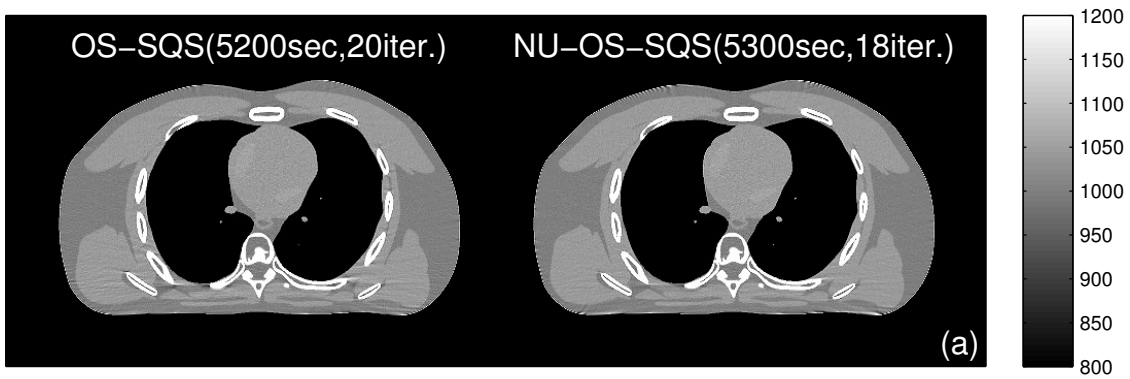
In Fig. 4.11(a), the reconstructed images of (NU-)OS-SQS show that NU method accelerates OS-SQS and reaches closer to the converged image after the same computation time (88 min.). This is apparent when comparing the difference images between the reconstructed and converged images in Fig. 4.11(b), particularly around the spine.

4.3 Other variation of SQS methods

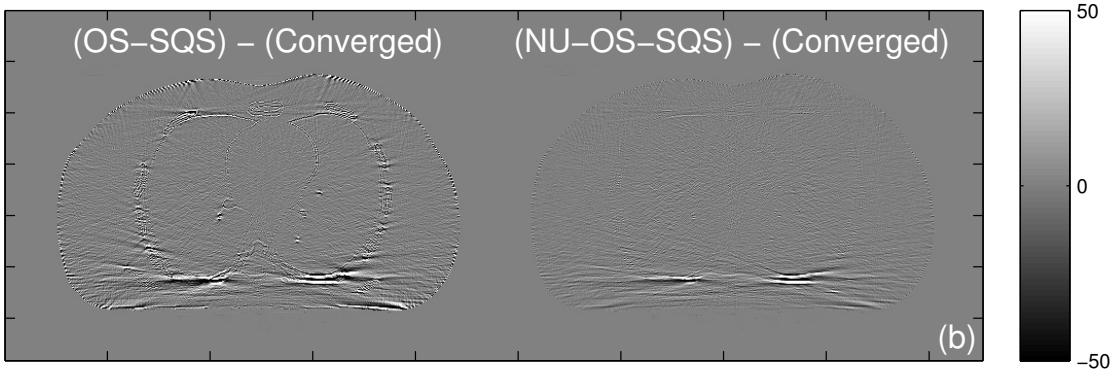
In addition to a promising NU-SQS method, we have also attempted to design SQS in other ways that lead to faster convergence rate than the standard SQS; a SQS with bounded interval (SQS-BI) for regularizer and a quasi-separable quadratic surrogates (QSQS).

4.3.1 SQS with bounded interval (SQS-BI) for regularizer $R(x)$

We present a new monotonic algorithm that is derived using SQS. The new algorithm accelerates the convergence rate by adapting reduced curvature values for the regularizer that



(a)



(b)

Figure 4.11: Simulated XCAT phantom: (a) Center slice of reconstructed image by OS-SQS(82) and NU-OS-SQS(82)-$g(v) = \max\{v^{10}, 0.05\}$ after about 88 min. (b) Difference between the reconstructed and converged images are additionally shown to illustrate the acceleration of NU approach. (Images are cropped for better visualization.)

were proposed by Yu *et al.* [109] for coordinate descent algorithms. We further accelerate the new algorithm by modifying Yu *et al.*'s curvature, since improvement using the original Yu *et al.*'s curvature is relatively small.

We follow the derivation of SQS in Section 4.1 but in an opposite order that first constructs a separable surrogate and designs a new tighter quadratic surrogate for the separable surrogate with respect to a bounded interval that is known to include the minimizer $\hat{x}$. So, the construction of a quadratic surrogate is different from the standard SQS.

4.3.1.1 SQS-BI algorithm for regularizer

We first derive a separable surrogate $S_R^{(n)}(x)$ for the regularizer $R(x)$ given in (2.8), using the convexity of $\psi_k(t)$ and the coefficient $\pi_{kj} = \frac{|c_{kj}|}{\sum_{l=1}^{N_p} |c_{rl}|}$ for the regularizer version of (4.8)

and (4.9):

$$R(x) \le S_R^{(n)}(x) \triangleq \sum_{j=1}^{N_p} S_{R,j}^{(n)}(x_j) \tag{4.46}$$

$$S_{R,j}^{(n)}(x_j) \triangleq \sum_{k=1}^{N_r} \pi_{kj} \psi_k \left(\frac{c_{kj}}{\pi_{kj}} (x_j - x_j^{(n)}) + [Cx^{(n)}]_k \right) = \sum_{k=1}^{N_r} \rho_{kj} \Big(x_j - r_{kj}^{(n)} \Big), \tag{4.47}$$

where $\rho_{kj}(t) \triangleq \pi_{kj} \psi_k \Big(\frac{c_{kj}}{\pi_{kj}} t \Big)$, and $r_{kj}^{(n)} \triangleq x_j^{(n)} - \frac{\pi_{kj}}{c_{kj}} [Cx^{(n)}]_k$.

Typically the regularization matrix C is sparse, so we can save computation by using the set $N_j \triangleq \{k = 1, \ldots, N_r : c_{kj} \neq 0\}$. Combining with the data-fit SQS surrogate in (4.9) yields the overall separable surrogate

$$S_j^{(n)}(x_j) \triangleq \phi_{L,j}^{(n)}(x_j) + S_{R,j}^{(n)}(x_j) = \phi_{L,j}^{(n)}(x_j) + \sum_{k \in N_j} \rho_{kj}(x_j - r_{kj}^{(n)}). \tag{4.48}$$

The separable surrogate $S_R^{(n)}(x)$ is not quadratic, so we design a quadratic surrogate next. The simplest design would be to find an upper bound on the curvature of $S_{R,j}^{(n)}(x_j)$, called the maximum curvature. Much smaller curvatures are derived in [51, Lemma 8.3, p.184] that are optimal when minimizing over the entire real line (see (4.54) below), called Huber's curvature. However, the minimizer of a separable surrogate always lies in a finite interval, and this property provides the opportunity to use the method in [109] that yields even smaller curvatures that can accelerate the convergence rate.

We assume that the potential function $\psi_k(t)$ satisfies the conditions in [109, Theorem 1], then $\rho_{kj}(t)$ also satisfies these. Then, the quadratic surrogate function for $\rho_{kj}(t)$ can be defined as

$$q_{kj}^{(n)}(t) \triangleq \rho_{kj}(\tilde{t}_{kj}^{(n)}) + \dot{\rho}_{kj}(\tilde{t}_{kj}^{(n)})(t - \tilde{t}_{kj}^{(n)}) + \frac{1}{2} \check{c}_{kj}^{(n)} (t - \tilde{t}_{kj}^{(n)})^2, \tag{4.49}$$

where $\tilde{t}_{kj}^{(n)} \triangleq x_j^{(n)} - r_{kj}^{(n)}$. We design the regularizer surrogate curvature $\check{c}_{kj}^{(n)}$ so that the surrogate $q_{kj}^{(n)}(t)$ satisfies the following conditions, which is modified from (2.15):

$$\begin{aligned} \rho_{kj}(x_j^{(n)} - r_{kj}^{(n)}) &= q_{kj}^{(n)}(x_j^{(n)} - r_{kj}^{(n)}) \\ \rho_{kj}(x_j - r_{kj}^{(n)}) &\le q_{kj}^{(n)}(x_j - r_{kj}^{(n)}), \quad \forall x_j \in P_j^{(n)}, \end{aligned} \tag{4.50}$$

where $P_j^{(n)} \triangleq \Big[p_j^{\min,(n)},\ p_j^{\max,(n)} \Big]$ is an interval containing the minimizer of $S_j^{(n)}(x_j)$. (Using the bounded interval $P_j^{(n)}$ instead of $\mathbb{R}_+^{N_p}$ in (2.15) is the main contribution here.) The set

$P_j^{(n)}$ is computed by finding the smallest and the largest minimizer of the convex functions

$$\left\{\phi_{L,j}^{(n)}(x_j),\ \rho_{kj}(x_j - r_{kj}^{(n)}),\ k \in N_j\right\} \tag{4.51}$$

The minimizers of the set (4.51) are $p_j^{(n)} \triangleq \arg\min_{x_j} \phi_{L,j}^{(n)}(x_j) = x_j^{(n)} - \frac{1}{d_j^{L,(n)}} \frac{\partial}{\partial x_j} L(x^{(n)})$, and $r_{kj}^{(n)} = \arg\min_{x_j} \rho_{kj}(x_j - r_{kj}^{(n)})$. Then we define $p_j^{\min,(n)}$ and $p_j^{max,(n)}$ to be the minimum and maximum of $\left\{u_j^{(n)}, r_{kj}^{(n)}, k \in N_j\right\}$. (We set $p_j^{\min,(n)}$ to be zero for nonnegativity constraint if the minimum of $\left\{u_j^{(n)}, r_{kj}^{(n)}, k \in N_j\right\}$ is negative.) Having the specified $P_j^{(n)}$, the optimal curvature $\check{c}_{kj}^{(n)}$ of $q_{kj}^{(n)}(t)$ satisfying the conditions (4.50) is computed by the method in [109, Fig. 12] using:

$$t_{kj}^{(n)} \triangleq \begin{cases} -\tilde{t}_{kj}^{(n)}, & |\tilde{t}_{kj}^{(n)}| \le \min\{|\tilde{t}_{kj}^{\min,(n)}|, |\tilde{t}_{kj}^{\max,(n)}|\} \\ \tilde{t}_{kj}^{\min,(n)}, & |\tilde{t}_{kj}^{\min,(n)}| \le \min\{|\tilde{t}_{kj}^{(n)}|, |\tilde{t}_{kj}^{\max,(n)}|\} \\ \tilde{t}_{kj}^{\max,(n)}, & \text{otherwise,} \end{cases} \tag{4.52}$$

where $\tilde{t}_{kj}^{\min,(n)} \triangleq p_j^{\min,(n)} - r_{kj}^{(n)}$, and $\tilde{t}_{kj}^{\max,(n)} \triangleq p_j^{\max,(n)} - r_{kj}^{(n)}$. Then, the optimal curvature $\check{c}_{kj}^{(n)}$, referred as Yu *et al.*'s curvature, which is the smallest curvature of $q_{kj}^{(n)}(t)$ satisfying the conditions (4.50), is

$$\check{c}_{kj}^{(n)} \triangleq \begin{cases} 2\left(\frac{\rho_{kj}(t_{kj}^{(n)}) - \rho_{kj}(\tilde{t}_{kj}^{(n)})}{(t_{kj}^{(n)} - \tilde{t}_{kj}^{(n)})^2} - \frac{\dot{\rho}_{kj}(\tilde{t}_{kj}^{(n)})}{t_{kj}^{(n)} - \tilde{t}_{kj}^{(n)}}\right), & \tilde{t}_{kj}^{(n)} \neq 0 \\ \ddot{\rho}_{kj}(0), & \text{otherwise.} \end{cases} \tag{4.53}$$

The traditional Huber's curvature is found by computing $\check{c}_{kj}^{(n)}$ for $t_{kj}^{(n)} = -\tilde{t}_{kj}^{(n)}$ where $P_j^{(n)} = (-\infty, \infty)$, and turns out to be

$$\check{c}_{kj}^{(n)} = \frac{\dot{\rho}_{kj}(\tilde{t}_{kj}^{(n)})}{\tilde{t}_{kj}^{(n)}}, \tag{4.54}$$

which is larger than Yu's optimal curvature (4.53).

The quadratic surrogate function of the separable surrogate $S_{R,j}^{(n)}(x_j)$ in (4.47) can be defined using (4.49) as

$$\phi_{R,BI,j}^{(n)}(x_j) \triangleq \sum_{k \in N_j} q_{kj}^{(n)}(x_j - r_{kj}^{(n)}). \tag{4.55}$$

Then the overall separable quadratic surrogate function for $R(x)$ can be written as follows:

$$\phi_{R,BI}^{(n)}(x) \triangleq R(x^{(n)}) + \nabla R(x^{(n)})(x - x^{(n)}) + \frac{1}{2}(x - x^{(n)})' \,\mathsf{diag}\Big\{d_j^{R,BI,(n)}\Big\}(x - x^{(n)}). \quad (4.56)$$

where $d_j^{R,BI,(n)} \triangleq \frac{\partial^2}{\partial x_j^2}\phi_{R,BI}^{(n)}(x) = \sum_{k \in N_j} \check{c}_{kj}^{(n)}$.

To summarize the surrogate derivations above:

$$\begin{aligned}\Psi(x) &\le \phi_{BI}^{(n)}(x) \triangleq \phi_L^{(n)}(x) + \phi_{R,BI}^{(n)}(x) \\ &= \Psi(x^{(n)}) + \nabla\Psi(x^{(n)})(x - x^{(n)}) + \frac{1}{2}(x - x^{(n)})' \,\mathsf{diag}\Big\{d_j^{BI,(n)}\Big\}(x - x^{(n)}),\end{aligned} \quad (4.57)$$

where $d_j^{BI,(n)} \triangleq d_j^{L,(n)} + d_j^{R,BI,(n)}$. Because $\phi_{BI}^{(n)}(x)$ is separable and quadratic, its unconstrained minimizer is easily derived to be:

$$\hat{p}_j^{(n)} \triangleq \arg\min_p \phi_j^{BI,(n)}(p) = x_j^{(n)} - \frac{1}{d_j^{BI,(n)}} \frac{\partial}{\partial x_j}\Psi(x^{(n)}). \quad (4.58)$$

However, $x_j^{(n+1)}$ must be in the set $P_j^{(n)}$ to maintain the monotonicity, so we revise (4.58) to be:

$$x_j^{(n+1)} = \arg\min_{p \in P_j^{(n)}} \phi_j^{BI,(n)}(p) = \mathrm{clip}\left\{\hat{p}_j^{(n)}, P_j^{(n)}\right\}, \quad (4.59)$$

where the clip function chooses a nearest value in the interval $P_j^{(n)}$ when $\hat{p}_j^{(n)}$ falls out of the interval.

We further accelerate the algorithm while preserving monotonicity by reducing the interval $P_j^{(n)}$ which in turn decreases the the optimal curvature (4.53). Define

$$\bar{P}_j^{(n)} = \begin{cases} \left[x_j^{(n)} - \eta\left(x_j^{(n)} - p_j^{\min,(n)}\right),\ x_j^{(n)} + \eta\left(p_j^{\max,(n)} - x_j^{(n)}\right)\right], & x_j^{(n)} \in P_j^{(n)} \\ P_j^{(n)}, & \text{otherwise,} \end{cases} \quad (4.60)$$

where $\eta \in [0, 1]$ is a user-selected reduction factor. The proposed curvature is computed by replacing $P_j^{(n)}$ by the reduced interval $\bar{P}_j^{(n)}$ in (4.60), and thereby the proposed algorithm is expected to converge faster. However, too much reduction can confine the update to a small interval, possibly slowing convergence.

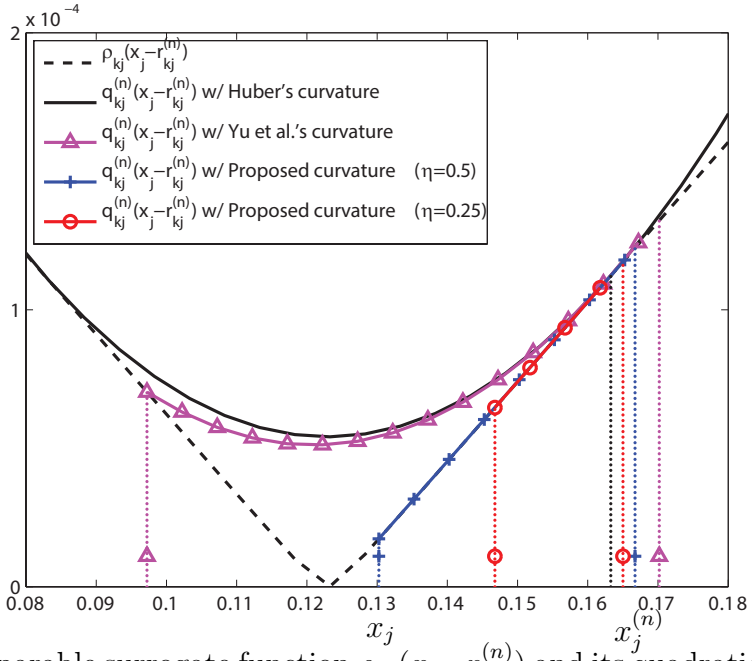


Figure 4.12: A separable surrogate function $\rho_{kj}(x_j - r_{kj}^{(n)})$ and its quadratic surrogate function $q_{kj}^{(n)}(x_j - r_{kj}^{(n)})$ with Huber's, Yu *et al.*'s and proposed curvatures. Proposed curvature is much smaller than Huber's curvature, while Yu *et al.*'s curvature is similar to Huber's curvature.

4.3.1.2 Simulation results

We evaluated the algorithm using a simulation data with the phantom in Fig. 4.13(a). The projection space is 444 detector elements and 20 projection views, and the reconstructed image is of size 256×256. The noisy sinogram data y was generated by the Poisson noise model (2.5). The FBP reconstructed image in Fig. 4.13(b) was used as the initial guess $x^{(0)}$ for the iterative reconstruction. We minimized the PWLS cost function $\Psi(x)$ with the statistical weighting $w_i = \exp(-y_i) \propto \frac{1}{\mathrm{var}(y_i)}$. The edge-preserving hyperbola potential function $\psi(t)$ (2.12) with $\delta = 0.005$, the regularization parameter $\beta_k = 0.25$ for horizontal and vertical differences, and $\beta_k = \frac{0.25}{\sqrt{2}}$ for diagonal differences was empirically chosen to produce a good image, where differencing matrix C had 1st-order differences in 2D. We reconstructed an image for each $\eta = 0.5$, 0.25, 0.125 and 1, 2 and 4 ordered subsets, with a nonnegativity constraint. The OS-SQS and OS-SQS-BI reconstructed images in Fig. 4.13 suggest that iterative image reconstruction can produce better images than FBP.

The proposed method uses smaller curvatures than previous work, thus converging faster than the conventional OS approach. Fig. 4.14 plots normalized root-mean squared difference (NRMSD) [dB]:

$$\mathrm{NRMSD} \triangleq 20 \log_{10} \left(\frac{\left|\left| x^{(n)} - \hat{x} \right|\right|_2}{\left|\left| \hat{x} \right|\right|_2} \right) \text{ [dB]} \tag{4.61}$$

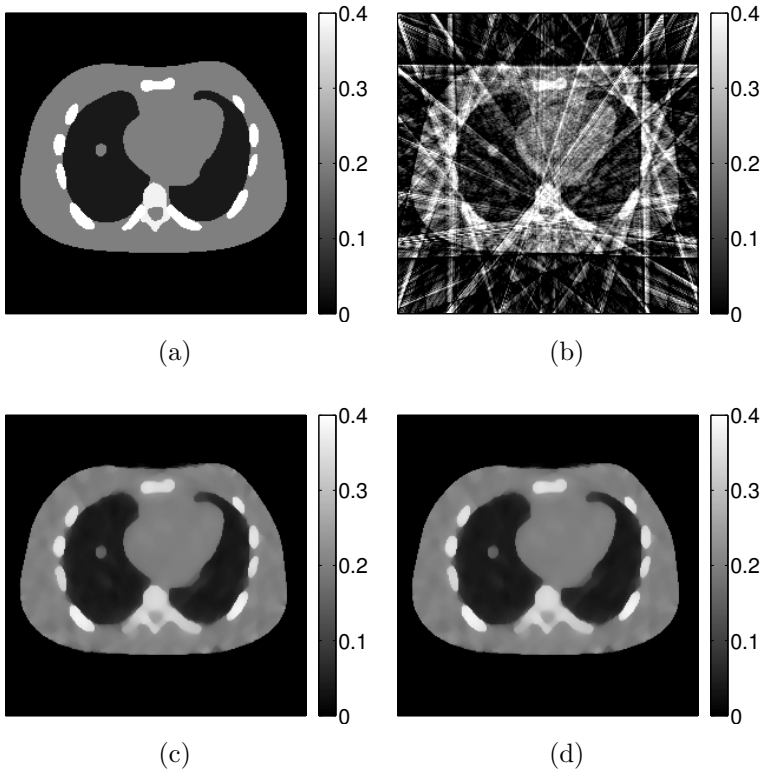


Figure 4.13: (a) Phantom image, (b) FBP image $x^{(0)}$, (c) OS-SQS image $x^{(330)}$ and (d) OS-SQS-BI image $x^{(290)}$ with $\eta = 0.25$ for 4 ordered subsets. The NRMSD for both (c) and (d) are -30 [dB].

between the current image $x^{(n)}$ and the converged image $\hat{x}$ (computed by 3000 iterations of SQS method), versus iteration. (The NRMSD plot (or RMSD [HU] plot) versus computation time would be useful for comparison, and we plan to acquire the data in near future.) The results show that OS-SQS-BI is about 15% faster than the standard OS-SQS. This is a modest improvement, but the extra curvature computation required in (4.53) is small compared to forward projection. Fig. 4.14 illustrates that both reducing the interval $P_j^{(n)}$ and using ordered subsets accelerate the SQS algorithm, as expected. However, the results show that reducing the interval too much slows down the convergence speed. We found that for densely sampled view angles, the acceleration was less significant because the reduced curvatures in $R(x)$ are overwhelmed by the curvature of $L(x)$.

4.3.2 Quasi-separable quadratic surrogates (QSQS)

We propose a quasi-separable quadratic surrogate (QSQS) algorithm that leads to a two-voxel-wise quasi-separable surrogate function with a *tridiagonal* Hessian matrix, where a

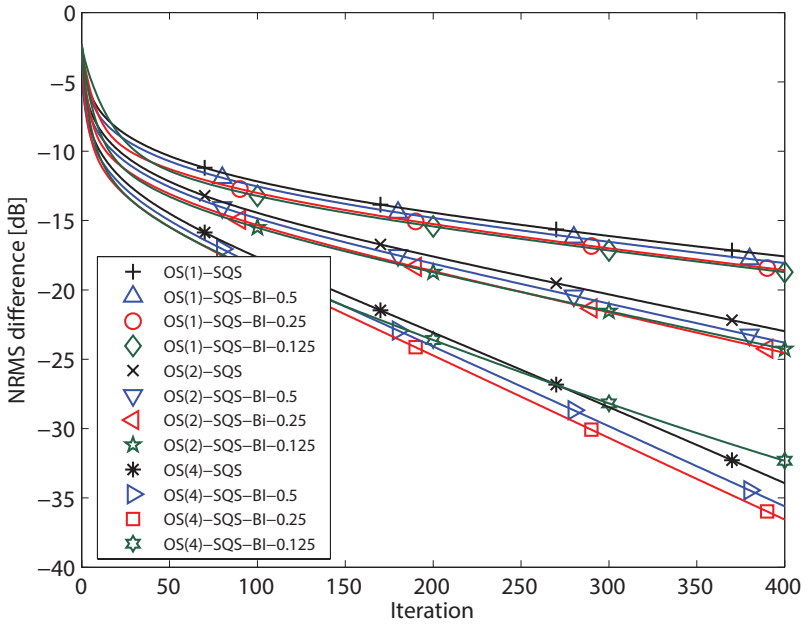


Figure 4.14: NRMSD [dB] versus iterations of OS-SQS and OS-SQS-BI with $\eta = 0.5$, 0.25, 0.125 for 1, 2 and 4 ordered subsets

tridiagonal Hessian matrix that can be inverted quickly [47, Chapter 5] [106], almost as fast as inverting a diagonal Hessian matrix. Therefore, we expect to have faster convergence in run time with a QSQS algorithm as it has smaller curvature than that of SQS while the computation time per iteration remains almost the same. In 2D X-ray CT, the QSQS approach will update either the horizontal or vertical profiles faster than SQS, depending on the ordering of x. (We have not extended this algorithm to 3D geometry, which we leave as possible future work if time permits.) We propose reordering of x at each iteration to encourage equivalent convergence rates for both horizontal and vertical profiles, which is also recommended in coordinate descent algorithm [96]. We further accelerate the algorithm using ordered subsets. Simulation results show that the proposed OS-QSQS algorithm converges faster than the OS-SQS algorithm for 2D X-ray CT reconstruction.

4.3.2.1 QSQS algorithm

We extend the derivation of SQS in Section 4.1. We first group each two adjacent voxels in order with a parameter $\theta_{ij} \in (0\ 1)$ for all i and j to determine the portion of grouping of

voxels, and illustrate the projection $[Ax]_i$ as follows:

$$[Ax]_i = \sum_{j=1}^{N_p} a_{ij}x_j = \sum_{j=1}^{N_p} \theta_{ij}a_{ij}x_j + \sum_{j=1}^{N_p}(1-\theta_{ij})a_{ij}x_j \tag{4.62}$$

$$= \sum_{j=0}^{N_p} \left(\theta_{ij}a_{ij}x_j + (1-\theta_{i,j+1})a_{i,j+1}x_{j+1}\right), \tag{4.63}$$

where we let $a_{ij} = 0$ and $x_j = 0$ for all i and $j \in \{0, N_p+1\}$ for simplicity. Then, we rewrite (4.63) using the idea of separable surrogate:

$$[Ax]_i = \sum_{j=0}^{N_p} \tilde{\pi}_{ij} \left(\frac{\theta_{ij}a_{ij}}{\tilde{\pi}_{ij}}(x_j - x_j^{(n)}) + \frac{(1-\theta_{i,j+1})a_{i,j+1}}{\tilde{\pi}_{ij}}(x_{j+1} - x_{j+1}^{(n)}) + [Ax^{(n)}]_i \right), \tag{4.64}$$

where $\sum_{j=0}^{N_p} \tilde{\pi}_{ij} = 1$ and $\tilde{\pi}_{ij}$ is nonnegative and zero only if $a_{ij} = 0$ and $a_{i,j+1} = 0$.

We construct a quadratic surrogate $Q_L^{(n)}(x) = \sum_{i=1}^{N_d} q_i^{(n)}([Ax]_i)$ for $L(x)$ as in (4.3), and derive the following by using the convexity inequality:

$$q_i^{(n)}([Ax]_i) \le \sum_{j=0}^{N_p} \tilde{\pi}_{ij} q_i^{(n)} \left(\frac{\theta_{ij}a_{ij}}{\tilde{\pi}_{ij}}(x_j - x_j^{(n)}) + \frac{(1-\theta_{i,j+1})a_{i,j+1}}{\tilde{\pi}_{ij}}(x_{j+1} - x_{j+1}^{(n)}) + [Ax^{(n)}]_i \right). \tag{4.65}$$

Then, we get a quasi-separable quadratic surrogate $\tilde{\phi}_L^{(n)}(x)$ for $L(x)$:

$$L(x) \le \tilde{\phi}_L^{(n)}(x) = \sum_{j=0}^{N_p} \tilde{\phi}_{L,j}^{(n)}(x_j, x_{j+1}) \tag{4.66}$$

$$\tilde{\phi}_{L,j}^{(n)}(x_j, x_{j+1}) \triangleq \sum_{i=1}^{N_d} \tilde{\pi}_{ij} q_i^{(n)} \left(\frac{\theta_{ij}a_{ij}}{\tilde{\pi}_{ij}}(x_j - x_j^{(n)}) + \frac{(1-\theta_{i,j+1})a_{i,j+1}}{\tilde{\pi}_{ij}}(x_{j+1} - x_{j+1}^{(n)}) + [Ax^{(n)}]_i \right). \tag{4.67}$$

The second derivatives of $\tilde{\phi}_L^{(n)}(x)$ are:

$$T_{jl}^{L,(n)} \triangleq \frac{\partial^2}{\partial x_j \partial x_l}\tilde{\phi}_L^{(n)}(x) = \begin{cases} \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \left(\frac{(1-\theta_{ij})^2}{\tilde{\pi}_{i,j-1}} + \frac{\theta_{ij}^2}{\tilde{\pi}_{ij}} \right) a_{ij}^2, & l = j \\ \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \frac{\theta_{ij}a_{ij}(1-\theta_{i,j+1})a_{i,j+1}}{\tilde{\pi}_{ij}}, & l = j+1 \\ 0, & \text{otherwise} \end{cases}, \tag{4.68}$$

where $\left\{\breve{c}_i^{(n)}\right\}$ are curvatures for the function $q_i^{(n)}(t)$. This reduces to the standard SQS choice

in (4.10) when $\theta_{ij} = 1$ for all i and j.

We select

$$\tilde{\pi}_{ij} \triangleq \frac{\theta_{ij}a_{ij} + (1-\theta_{i,j+1})a_{i,j+1}}{\sum_{l=1}^{N_p} a_{il}},$$

where $a_{ij} \geq 0$ for all i and j here, but any other choice of $\tilde{\pi}_{ij}$ satisfying its conditions can be used. In particular, the nonuniform ideas of Section 4.2.2 could also be used here. For this choice, the surrogate Hessian becomes

$$\frac{\partial^2}{\partial x_j \partial x_l}\tilde{\phi}_L^{(n)}(x) = \begin{cases} \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \left(\frac{(1-\theta_{ij})^2}{\theta_{i,j-1}a_{i,j-1}+(1-\theta_{ij})a_{ij}} + \frac{\theta_{ij}^2}{\theta_{ij}a_{ij}+(1-\theta_{i,j+1})a_{i,j+1}} \right) a_{ij}^2 \sum_{l=1}^{N_p} a_{il}, & l = j \\ \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \frac{\theta_{ij}a_{ij}(1-\theta_{i,j+1})a_{i,j+1}}{\theta_{ij}a_{ij}+(1-\theta_{i,j+1})a_{i,j+1}} \sum_{l=1}^{N_p} a_{il}, & l = j+1 \\ 0, & \text{otherwise.} \end{cases} \tag{4.69}$$

We choose $\theta_{ij} = \frac{1}{2}$ for all i and j for simplicity, which leads to uniform grouping. (Other choices of θ_{ij} can be possible but we will not discuss here.) This choice leads to the following expression for the elements of the tridiagonal Hessian matrix:

$$\frac{\partial^2}{\partial x_j \partial x_l}\tilde{\phi}_L^{(n)}(x) = \begin{cases} \frac{1}{2} \sum_{i \in \tilde{N}_j} \breve{c}_i^{(n)} \left(\frac{1}{a_{i,j-1}+a_{ij}} + \frac{1}{a_{ij}+a_{i,j+1}} \right) a_{ij}^2 \sum_{l=1}^{N_p} a_{il}, & l = j \\ \frac{1}{2} \sum_{i \in \tilde{N}_j \cap \tilde{N}_{j+1}} \breve{c}_i^{(n)} \frac{a_{ij}a_{i,j+1}}{a_{ij}+a_{i,j+1}} \sum_{l=1}^{N_p} a_{il}, & l = j+1 \\ 0, & \text{otherwise} \end{cases}, \tag{4.70}$$

where $\tilde{N}_j \triangleq \{i = 1, \ldots, N_d \,:\, a_{ij} \neq 0\}$. The complicated form (4.70) for the tridiagonal Hessian can be simplified as follows:

$$\frac{\partial^2}{\partial x_j \partial x_l}\tilde{\phi}_L^{(n)}(x) = \begin{cases} \sum_{i \in \tilde{N}_j} \breve{c}_i^{(n)} a_{ij} \sum_{l=1}^{N_p} a_{il} - \left(\frac{\partial^2}{\partial x_j \partial x_{j-1}}\tilde{\phi}_L^{(n)}(x) + \frac{\partial^2}{\partial x_j \partial x_{j+1}}\tilde{\phi}_L^{(n)}(x) \right), & l = j \\ \frac{1}{2} \sum_{i \in \tilde{N}_j \cap \tilde{N}_{j+1}} \breve{c}_i^{(n)} \frac{a_{ij}a_{i,j+1}}{a_{ij}+a_{i,j+1}} \sum_{l=1}^{N_p} a_{il}, & l = j+1 \\ 0, & \text{otherwise} \end{cases}, \tag{4.71}$$

which can be easily calculated by computing the Hessian of SQS (4.10) and additional off-diagonal elements of Hessian for QSQS.

The QSQS of $L(x)$ can be written as

$$\tilde{\phi}_L^{(n)}(x) \triangleq L(x^{(n)}) + \nabla L(x^{(n)})(x - x^{(n)}) + \frac{1}{2}(x - x^{(n)})' T^{L,(n)}(x - x^{(n)}), \tag{4.72}$$

where $T^{L,(n)} = \left\{T_{jl}^{L,(n)}\right\}_{N_p \times N_p}$ is a symmetric tridiagonal matrix.

Similarly, we can construct a QSQS surrogate $\tilde{\phi}_R^{(n)}(x)$ for the regularizer $R(x)$ with Hessian values:

$$T_{jl}^{R,(n)} \triangleq \frac{\partial^2}{\partial x_j \partial x_l} \tilde{\phi}_R^{(n)}(x) \tag{4.73}$$

$$= \begin{cases} \frac{1}{2} \sum_{k \in \tilde{K}_j} \ddot{\psi}_k(0) \left(\frac{1}{|c_{k,j-1}|+|c_{kj}|} + \frac{1}{|c_{kj}|+|c_{k,j+1}|} \right) c_{kj}^2 \sum_{l=1}^{N_p} |c_{rl}|, & l = j \\ \frac{1}{2} \sum_{k \in \tilde{K}_j \cap \tilde{K}_{j+1}} \ddot{\psi}_k(0) \frac{c_{kj} c_{k,j+1}}{|c_{kj}|+|c_{k,j+1}|} \sum_{l=1}^{N_p} |c_{rl}|, & l = j+1 \\ 0, & \text{otherwise} \end{cases} \tag{4.74}$$

$$= \begin{cases} \sum_{k \in \tilde{K}_j} \ddot{\psi}_k(0) c_{kj} \sum_{l=1}^{N_p} |c_{rl}| - \left(\frac{\partial^2}{\partial x_j \partial x_{j-1}} \tilde{\phi}_R^{(n)}(x) + \frac{\partial^2}{\partial x_j \partial x_{j+1}} \tilde{\phi}_R^{(n)}(x) \right), & l = j \\ \frac{1}{2} \sum_{k \in \tilde{K}_j \cap \tilde{K}_{j+1}} \ddot{\psi}_k(0) \frac{c_{kj} c_{k,j+1}}{|c_{kj}|+|c_{k,j+1}|} \sum_{l=1}^{N_p} |c_{rl}|, & l = j+1 \\ 0, & \text{otherwise} \end{cases}, \tag{4.75}$$

where $\tilde{K}_j = \{k = 1, \ldots, N_r \,:\, c_{kj} \neq 0\}$. (Compare (4.75) with the denominator of regularzier for the standard SQS in (4.11).)

The overall QSQS surrogate can be defined as:

$$\begin{aligned} \Psi(x) \leq \tilde{\phi}^{(n)}(x) &\triangleq \tilde{\phi}_L^{(n)}(x) + \tilde{\phi}_R^{(n)}(x) \\ &= \Psi(x^{(n)}) + \nabla \Psi\left(x^{(n)}\right)(x - x^{(n)}) + \frac{1}{2}(x - x^{(n)})' T^{(n)} (x - x^{(n)}), \end{aligned} \tag{4.76}$$

where $T^{(n)} \triangleq T^{L,(n)} + T^{R,(n)} = \left\{T_{jl}^{L,(n)} + T_{jl}^{R,(n)}\right\}_{N_p \times N_p}$ is a symmetric tridiagonal matrix. For computational efficiency, we precompute the coefficients that are used for computing $T^{L,(n)}$ and $T^{R,(n)}$ before starting the iterative algorithm, which takes longer than the precomputation needed for SQS. In addition, we need doubled memory space for precomputed $T^{(n)}$, compared to that of SQS, due to the off-diagonal elements in Hessian of QSQS. The following is the update equation of QSQS:

$$x^{(n+1)} = \arg\min_x \tilde{\phi}^{(n)}(x) = x^{(n)} - \left[T^{(n)}\right]^{-1} \nabla \Psi\left(x^{(n)}\right). \tag{4.77}$$

All voxels can be updated relatively fast, since $\left[T^{(n)}\right]^{-1} \nabla \Psi\left(x^{(n)}\right)$ can be computed in $O(n)$ operations instead of $O(n^3)$ that is required for a dense Hessian matrix [47, Chapter 5] [106]. This enables the QSQS algorithm to have similar computation for each iteration as SQS, while converging faster than SQS due to a smaller curvature for surrogate.

4.3.2.2 Reordering of x in horizontal and vertical direction

We expect (4.77) to update either the horizontal or vertical profiles faster than SQS, depending on the ordering of x. To accelerate convergence, we further suggest sequentially reordering x to update the horizontal and vertical profiles respectively. In [96], a related reordering idea is suggested to compensate the update-direction related convergence behavior of the coordinate descent algorithm. The pseudo code of the proposed algorithm is illustrated in Table 4.3. We expect the QSQS method will likely speed up the convergence for both horizontal and vertical direction, with a little extra computation and memory space. We refer to this algorithm as QSQS with reordering (QSQS-R).

1: Precompute the coefficients that is used for computing $T_h^{(n)}$ and $T_v^{(n)}$
2: for $n = 0, 1, 2, \cdots$
3: if n is odd
4: Order $x^{(n)}$ in horizontal directions: $x_h^{(n)}$
5: $x_h^{(n+1)} = \arg\min_{x_h} \tilde{\phi}_h^{(n)}(x_h) = x_h^{(n)} - \left[T_h^{(n)}\right]^{-1} \nabla \Psi_h \left(x_h^{(n)}\right)$
6: else
7: Order $x^{(n)}$ in vertical directions: $x_v^{(n)}$
8: $x_v^{(n+1)} = \arg\min_{x_v} \tilde{\phi}_v^{(n)}(x_v) = x_v^{(n)} - \left[T_v^{(n)}\right]^{-1} \nabla \Psi_v \left(x_v^{(n)}\right)$
9: end
10: end

Table 4.3: Pseudo code of QSQS algorithm with reordering of x in horizontal and vertical direction. (Each subscript h and v denote horizontal and vertical direction.)

4.3.2.3 Simulation results

The algorithm is evaluated with a simulation data using the phantom in Fig. 4.15(a). The projection space have 444 detector elements and 492 projection views, and the reconstructed image is of size 256×256. The noisy sinogram data y was generated by the Poisson noise model (2.5). The FBP reconstructed image in Fig. 4.15(b) was used for initializing the iterative reconstruction. We minimized PWLS cost function with statistical weighting parameter $w_i = \exp(-y_i) \propto \frac{1}{\mathrm{var}(y_i)}$. The edge-preserving hyperbola potential function (2.12) with $\delta = 0.005$, and the spatial weighting $\beta_k = 0.25$ for horizontal and vertical differences, and $\beta_k = \frac{0.25}{\sqrt{2}}$ for diagonal differences was empirically chosen to produce a good image, where differencing matrix C had 1st-order differences in 2D. We reconstructed an image for 1, 12 and 41 ordered subsets, with the nonnegativity constraint. The OS-SQS, OS-QSQS and

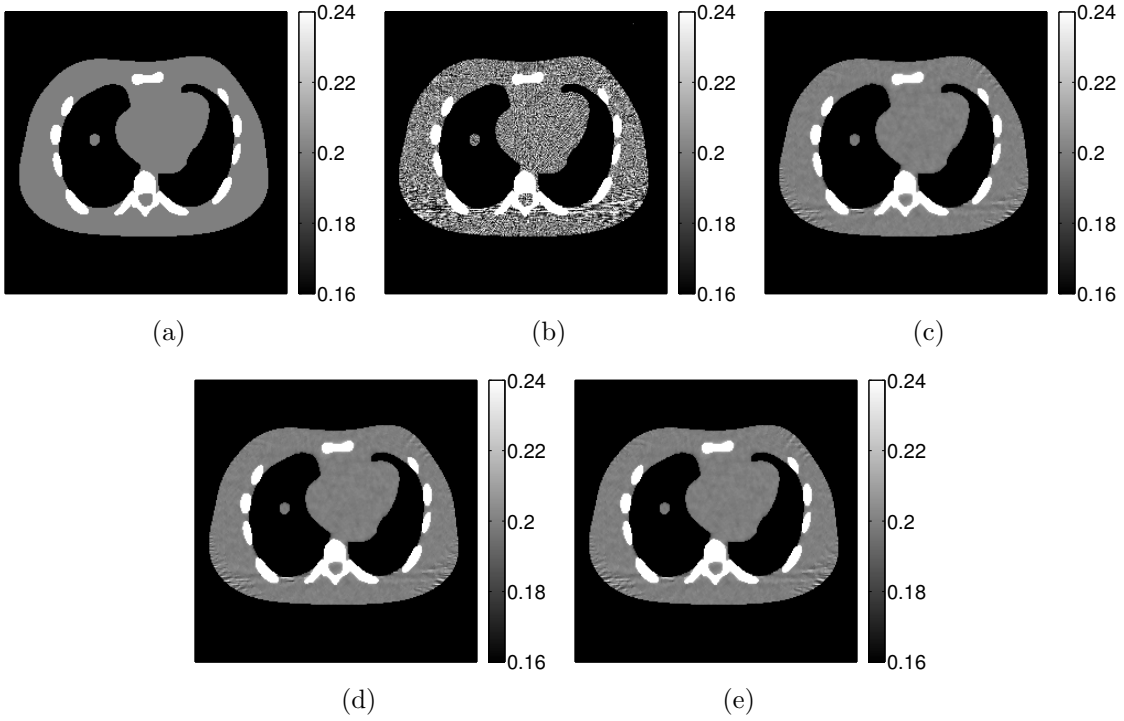


Figure 4.15: (a) Phantom image, (b) FBP image $x^{(0)}$, (c) OS-SQS image $x^{(34)}$, (d) OS-QSQS image $x^{(32)}$ and (e) OS-QSQS-R image $x^{(26)}$ for 41 ordered subsets. The NRMSD for (c), (d) and (e) are -40 [dB].

OS-QSQS-R reconstructed images in Fig. 4.15 suggest that iterative image reconstruction can produce better images than FBP.

The proposed method uses smaller curvatures than a standard SQS method, thus the proposed OS-QSQS converges faster than the OS-SQS algorithm. Fig. 4.16 plots NRMSD [dB] (4.61) between the current image $x^{(n)}$ and the converged image $\hat{x}$ (reconstructed by 3000 iterations of SQS), versus iteration. (The plot of NRMSD [dB] (or RMSD [HU]) versus computation time would be useful for comparison, which we leave it as a future work.) The results show that OS-QSQS-R is about 25% faster than the standard OS-SQS. This is a modest improvement, but the extra computation required for computing a tridiagonal Hessian matrix and inverting the matrix is small compared to forward projection. Fig. 4.16 illustrates that both reordering of x and using ordered subsets accelerate the QSQS algorithm, as expected. We leave the comparison of the proposed algorithms in computation time.

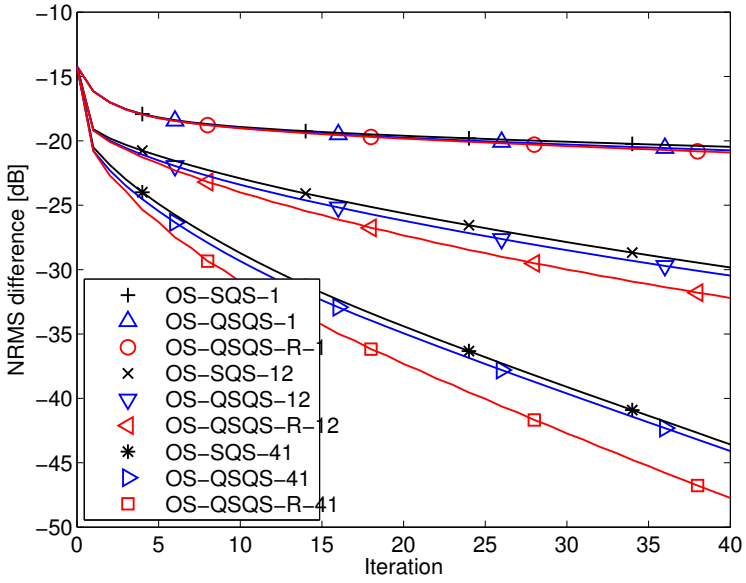


Figure 4.16: NRMSD [dB] versus iterations of OS-SQS, OS-QSQS and OS-QSQS-R for 1, 12 and 41 ordered subsets

4.4 Conclusion and Discussion

We have carefully studied the SQS method that is widely used in X-ray CT reconstruction, and proposed three novel approaches to improve the convergence rate of the SQS. We found the performance of NU-SQS algorithm the best among three approaches, which led us to further investigate NU-SQS method on 3D real patient data. In future work, we also plan to check SQS-BI and QSQS on real patient data set.

The key of the NU-SQS approach is designing "update-needed" factors $u_j^{(n)}$ in (4.22) that encourage larger step sizes for voxels that are predicted to need larger changes to reach the final image. Further optimization of these factors, *e.g.*, by improving the initialization of $\tilde{u}_j^{(0)}$ and the DRA function in (4.32), should lead to further acceleration and stability of the proposed NU-SQS and NU-OS-SQS methods.

CHAPTER V

Momentum approaches with ordered subsets

The accelerated optimization transfer methods with OS framework described in the previous chapter provided promising convergence rate but they are not yet fast enough. We provide further acceleration by introducing momentum approaches in Section 2.3.4 into OS-SQS methods. We call this combination the OS-momentum method. We particularly focus on Nesterov's two momentum methods [79, 83] in Section 2.3.4, which provide a fast convergence rate $O\left(\frac{1}{n^2}\right)$ where n counts the number of iterations. However, the combination of OS and momentum sometimes leads to instability. So, we adapted a relaxation scheme in [29] to stabilize the proposed OS-momentum algorithm.

Section 5.1 led to two conference papers [63, 64], and the initial version of Section 5.2 was presented in [58]. The journal version [65] of this chapter is currently in a review process.

5.1 OS-SQS methods with Nesterov's momentum

To further accelerate OS-SQS methods, we propose to adapt two of Nesterov's momentum techniques [79, 83]. (We can also consider another Nesterov's momentum approach [82] achieving same rate as other two [79, 83].) This section reviews both momentum approaches and combines them with OS methods.

The first momentum method [79] uses two previous iterates, while the second [83] accumulates all gradients. Without using OS methods, both Nesterov methods provide $O\left(1/n^2\right)$ convergence rates. We expect that combining momentum with OS methods will provide $O\left(1/(nM)^2\right)$ rates in early iterations. The main benefit of combining OS and Nesterov's momentum is that we have approximately M^2 times acceleration in early iterations with M subsets, while the extra computation and memory needed for momentum technique in OS-SQS are almost negligible. We discuss both proposed algorithms in more detail.

5.1.1 Proposed OS-SQS methods with momentum 1 (OS-mom1)

Table 5.1 illustrates the proposed combination of an OS-SQS algorithm with the momentum technique that is described in [79], where the algorithm generates two sequences $\{x^{(n+\frac{m}{M})}\}$ and $\{z^{(n+\frac{m}{M})}\}$, and line 7 of the algorithm corresponds to a momentum step with Nesterov's optimal parameter sequence t_k. Table 5.1 reduces to the ordinary OS-SQS algorithm in Table 2.3 when $t_k = 1$ for all $k \geq 0$.

1: Initialize $x^{(0)} = z^{(0)}$, $t_0 = 1$ and compute D.
2: for $n = 0, 1, \cdots, N-1$
3: for $m = 0, 1, \cdots, M-1$
4: $\quad k = nM + m$
5: $\quad t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$
6: $\quad x^{(\frac{k+1}{M})} = \left[z^{(\frac{k}{M})} - D^{-1}M\nabla\Psi_m(z^{(\frac{k}{M})})\right]_+$
7: $\quad z^{(\frac{k+1}{M})} = x^{(\frac{k+1}{M})} + \frac{t_k - 1}{t_{k+1}}\left(x^{(\frac{k+1}{M})} - x^{(\frac{k}{M})}\right)$
8: end
9: end

Table 5.1: Proposed OS-SQS methods with momentum in [79] (OS-mom1)

The non-OS version of Table 5.1 satisfies the following convergence rate:

Lemma 3. *For $n \geq 0$, the sequence $\left\{x^{(n)}\right\}$ generated by the non-OS version ($M = 1$) of Table 5.1 satisfies*

$$\Psi(x^{(n+1)}) - \Psi(\hat{x}) \leq \frac{2||x^{(0)} - \hat{x}||_D^2}{(n+1)(n+2)}, \tag{5.1}$$

where D is a majorizing matrix such as (2.19).

The inequality (5.1) is a simple generalization of [10, Theorem 4.4]. In practice, we expect the initial rate of OS-mom1 for $M > 1$ to have $O\left(1/(nM)^2\right)$ with the approximation (2.21), which is the main benefit of this work, while the computation cost remains almost the same as that of OS-SQS algorithm in Table 2.3. The only slight drawback of Table 5.1 over Table 2.3 is the extra memory needed to store the image z.

5.1.2 Proposed OS-SQS methods with momentum 2 (OS-mom2)

The second proposed OS-SQS algorithm with momentum in [83] is described in Table 5.2. The choice of coefficient t_k in Table 5.2 is adopted from [30, 104] and gives faster convergence than the choice in [83].

1: Initialize $x^{(0)} = v^{(0)} = z^{(0)}$, $t_0 = 1$ and compute D.
2: for $n = 0, 1, \cdots, N-1$
3: for $m = 0, 1, \cdots, M-1$
4: $\quad k = nM + m$
5: $\quad t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$
6: $\quad x^{(\frac{k+1}{M})} = \left[z^{(\frac{k}{M})} - D^{-1} M \nabla \Psi_m(z^{(\frac{k}{M})})\right]_+$
7: $\quad v^{(\frac{k+1}{M})} = \left[z^{(0)} - D^{-1} \sum_{l=0}^{nM+m} t_l M \nabla \Psi_{(l)_M}(z^{(\frac{l}{M})})\right]_+$
8: $\quad z^{(\frac{k+1}{M})} = x^{(\frac{k+1}{M})} + \frac{t_{k+1}}{\sum_{l=0}^{k+1} t_l}\left(v^{(\frac{k+1}{M})} - x^{(\frac{k+1}{M})}\right)$
9: end
10: end

Table 5.2: Proposed OS-SQS methods with momentum in [83] (OS-mom2), The notation $(l)_M$ denotes l mod M.

The sequence $\left\{x^{(n)}\right\}$ generated by Table 5.2 with $M = 1$ can be proven to satisfy the inequality (5.1), by generalizing [83, Theorem 2]. While the one-subset ($M = 1$) version of Table 5.2 provides $O(1/n^2)$, we expect from (2.21) for the OS version to have the rate $O\left(1/(nM)^2\right)$ in early iterations. Compared with Table 5.1, one additional $[\cdot]_+$ operation per iteration and extra arithmetic operations are required in Table 5.2, but those are negligible.

Overall, the two proposed algorithms in Tables 5.1 and 5.2 are expected to provide fast convergence rate $O\left(1/(nM)^2\right)$ in early iterations, which we confirm empirically in Section 5.3. However, the type of momentum affects the overall convergence when combined with OS. Also, the convergence behavior of two algorithms is affected by the number and ordering of subsets, as discussed in Section 5.3.

The proposed OS-momentum algorithms in Tables 5.1 and 5.2 become unstable in some cases, as predicted by the convergence analysis in [29]. To stabilize the algorithms, the next section proposes to adapt a recent relaxation scheme [29] developed for stochastic gradient methods with momentum.

5.2 Relaxation of momentum

This section relates the OS-SQS algorithm to *diagonally preconditioned* stochastic gradient methods and adapts a relaxation scheme designed for stochastic gradient algorithms with momentum. Then we investigate various choices of relaxation to achieve overall fast convergence.

5.2.1 Stochastic gradient method

One can view OS methods as stochastic gradient methods by defining $M\nabla\Psi_{S_k}(x)$ as a stochastic estimate of $\nabla\Psi(x)$, where a random variable S_k at kth iteration is uniformly chosen from $\{0, 1, \cdots, M-1\}$. In this stochastic setting, OS-SQS methods satisfy:

$$\begin{cases} \mathbb{E}\left[M\Psi_{S_k}(x)\right] = \Psi(x) \\ \mathbb{E}\left[M\nabla\Psi_{S_k}(x)\right] = \nabla\Psi(x) \\ \mathbb{E}\left[\left(M\nabla_j\Psi_{S_k}(x) - \nabla_j\Psi(x)\right)^2\right] \le \sigma_j^2, \ \forall j \end{cases} \tag{5.2}$$

for all $x \in \mathcal{B}$, for some finite constants $\{\sigma_j\}$, where $\mathbb{E}$ is the expectation operator over the random selection of S_k, $\nabla_j \triangleq \partial/\partial x_j$, and $\mathcal{B}$ is a bounded feasible set that includes $\hat{x}$. The feasible set $\mathcal{B}$ can be derived based on the measurement data y and the derivation in [4, Section A.2], and we reasonably assume that the sequences generated by the algorithms are within the set $\mathcal{B}$. The last inequality in (5.2) is a generalized version of [29, eqn. (2.5)] for (diagonally preconditioned) OS-SQS-type algorithms. The matrix $\Sigma \triangleq \mathsf{diag}\{\sigma_j\}$ has smaller values if we use smaller M and group the views into subsets appropriately, but it is impractical to compute Σ. (Section 5.3.1.4 provides a practical approach for approximating Σ.)

5.2.2 Proposed OS-SQS methods with relaxed momentum (OS-mom3)

Inspired by [29], Table 5.3 describes a generalized version of OS-SQS-momentum methods, which reduces to OS-mom2 algorithm in Table 5.2 with a deterministic subset ordering $S_k = (k \bmod M)$ and a fixed majorizing diagonal matrix

$$\Gamma^{(k)} = D. \tag{5.3}$$

For $M = 1$, the algorithm with these choices satisfies (5.1) with $n = k$. However, for $M > 1$, the analysis in [29] illustrates that using the choice (5.3) leads to the following inequality

$$\mathbb{E}\left[\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x})\right] \le \frac{2||x^{(0)} - \hat{x}||_D^2}{(k+1)(k+2)} + \frac{(k+3)\,\mathsf{tr}\{P\Sigma\}}{3} \tag{5.4}$$

for $k \ge 0$, where $P \triangleq \mathsf{diag}\Big\{p_j \triangleq \max_{x,\bar{x}\in\mathcal{B}} |x_j - \bar{x}_j|\Big\}$ measures the diameter of the feasible set $\mathcal{B}$. This expression reveals that OS methods with momentum may suffer from error accumulation due to the last term in (5.4) that depends on the error bounds (Σ) in (5.2). To improve stability, we would like to find a way to decrease this term. Using a larger constant

1: Initialize $x^{(0)} = v^{(0)} = z^{(0)}$, $t_0 \in (0,\ 1]$ and compute D.
2: for $n = 0, 1, \cdots, N-1$
3: for $m = 0, 1, \cdots, M-1$
4: $\quad k = nM + m$
5: $\quad$ Choose $\Gamma^{(k)} \triangleq \mathsf{diag}\left\{\gamma_j^{(k)}\right\}$ s.t. $\begin{cases}\Gamma^{(0)} \succ D, & k=0\\ \Gamma^{(k)} \succeq \Gamma^{(k-1)}, & k>0\end{cases}$
6: $\quad$ Choose t_{k+1} s.t. $t_{k+1}^2 \Gamma^{(k+1)} \preceq \left(\sum_{l=0}^{k+1} t_l\right) \Gamma^{(k)}$
7: $\quad x^{(\frac{k+1}{M})} = \left[z^{(\frac{k}{M})} - \left[\Gamma^{(k)}\right]^{-1} M \nabla \Psi_{\xi_k}(z^{(\frac{k}{M})})\right]_+$
8: $\quad v^{(\frac{k+1}{M})} = \left[z^{(0)} - \left[\Gamma^{(k)}\right]^{-1} \sum_{l=0}^{k} t_l M \nabla \Psi_{\xi_l}(z^{(\frac{l}{M})})\right]_+$
9: $\quad z^{(\frac{k+1}{M})} = x^{(\frac{k+1}{M})} + \frac{t_{k+1}}{\sum_{l=0}^{k+1} t_l}\left(v^{(\frac{k+1}{M})} - x^{(\frac{k+1}{M})}\right)$
10: end
11: end

Table 5.3: Proposed stochastic OS-SQS algorithms with momentum (OS-mom3). ξ_k is a realization of a random variable S_k.

denominator, *i.e.* $\Gamma^{(k)} = qD$ for $q > 1$, only slows down the accumulation of error and does not prevent eventual accumulation of error [29].

To stabilize the algorithm, we adapt the *relaxed* momentum approach in [29] as described in Table 5.3 with appropriately selected $\Gamma^{(k)}$ and t_k. Then, the algorithm in Table 5.3 satisfies the following convergence rate:

Lemma 4. *For $k \geq 0$, the sequence $\{x^{(\frac{k+1}{M})}\}$ generated by Table 5.3 satisfies*

$$\mathbb{E}\left[\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x})\right] \leq \frac{1}{\sum_{l=0}^{k} t_l}\left[\frac{||x^{(0)} - \hat{x}||^2_{\Gamma^{(k)}}}{2} + \sum_{l=0}^{k}\sum_{i=0}^{l} t_i (\Gamma^{(k)} - D)^{-1}\Sigma^2\right]. \tag{5.5}$$

Proof. See Appendix B. □

Lemma 4 shows that increasing $\Gamma^{(k)}$ can help prevent accumulation of error Σ. Next we discuss the selection of parameters $\Gamma^{(k)}$ and t_k.

5.2.3 The choice of $\Gamma^{(k)}$ and t_k

For any given $\Gamma^{(k)} \triangleq \mathsf{diag}\left\{\gamma_j^{(k)}\right\}$, we use $t_0 = 1$ and the following rule:

$$t_{k+1} = \frac{1}{2\alpha^{(k+1)}}\left(1 + \sqrt{1 + 4t_k^2 \alpha^{(k)}\alpha^{(k+1)}}\right) \tag{5.6}$$

for all $k \geq 0$, where $\alpha^{(k+1)} \triangleq \max_j \left(\gamma_j^{(k+1)}/\gamma_j^{(k)}\right)$ and $\alpha^{(0)} = 1$. The choice (5.6) increases the fastest among all possible choices satisfying the condition in line 6 of Table 5.3 (see the proof in Appendix C).[1]

Here, we focus on the choice

$$\Gamma^{(k)} = D + (k+2)^{c^{(k)}}\,\Gamma \tag{5.7}$$

for a nondecreasing $c^{(k)} \geq 0$ and a fixed diagonal matrix $\Gamma \triangleq \mathsf{diag}\{\gamma_j\} \succ 0$. The choice (5.7) is a generalized version of the work in [29], enabling more flexibility in the choice of $c^{(k)}$. We leave other formulations of $\Gamma^{(k)}$ that may provide better convergence as future work.

For $\Gamma^{(k)}$ in (5.7), computing $\alpha^{(k)}$ in (5.6) becomes

$$\alpha^{(k+1)} = 1 + \frac{(k+3)^{c^{(k+1)}} - (k+2)^{c^{(k)}}}{\min_j(d_j/\gamma_j) + (k+2)^{c^{(k)}}}. \tag{5.8}$$

Overall, the computational cost of Table 5.3 with the choices (5.6) and (5.7) remains similar to that of Table 5.2. Using (5.6) and (5.7), the proposed algorithm in Table 5.3 achieves the following inequality:

Corollary 2. *For $k \geq 0$, the sequence $\{x^{(\frac{k+1}{M})}\}$ generated by Table 5.3 with the coefficients* (5.6) *and* (5.7) *satisfies*

$$\mathbb{E}\left[\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x})\right] \leq \left(\max_{0\leq l\leq k} \sqrt{\alpha^{(l)}}\right)\left[\frac{2||x^{(0)} - \hat{x}||_D^2}{(k+1)(k+2)}\right.$$
$$\left. + \frac{2||x^{(0)} - \hat{x}||_\Gamma^2}{(k+1)(k+2)^{1-c^{(k)}}} + \frac{2\sum_{i=0}^{k}(i+2)^{2-c^{(i)}}\,\mathsf{tr}\{\Gamma^{-1}\Sigma^2\}}{(k+1)(k+2)}\right]. \tag{5.9}$$

Proof. Use Lemma 4 and the inequality conditions of the sequence $\{t_k\}$ in (5.6):

$$\frac{k+1}{2\sqrt{\alpha^{(k)}}} \leq t_k \leq \frac{k+1}{\sqrt{\alpha^{(k)}}},$$
$$\min_{0\leq l\leq k} \frac{(k+1)(k+2)}{4\sqrt{\alpha^{(l)}}} \leq \sum_{l=0}^{k} t_l \leq \max_{0\leq l\leq k} \frac{(k+1)(k+2)}{2\sqrt{\alpha^{(l)}}}$$

for $k \geq 0$, which can be easily proven by induction. □

There are two parameters $c^{(k)}$ and Γ to be tuned in (5.7). Based on Corollary 2, the next two subsections explore how these parameters affect convergence rate. (We made a

[1] The coefficient (5.6) increases faster than the choice $t_{k+1} = \frac{k+1}{2^c}$ for a constant $c^{(k)} = c \geq 1$ used in [29], so we use the choice (5.6) that leads to faster convergence based on Lemma 4.

preliminary investigation of these two parameters in [58].)

5.2.4 The choice of $c^{(k)}$

In Corollary 2, the choice of $c^{(k)}$ controls the overall convergence rate. We first consider a constant $c^{(k)}$.

Corollary 3. *For $k \geq 0$ and a fixed constant $c^{(k)} = c \in [0, 2]$, the sequence $\{x^{(\frac{k+1}{M})}\}$ generated by Table 5.3 with the coefficients* (5.6) *and* (5.7) *satisfies*

$$\mathbb{E}\left[\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x})\right] \leq \left(\max_{0\leq l\leq k} \sqrt{\alpha^{(l)}}\right) \left[\frac{2||x^{(0)} - \hat{x}||_D^2}{(k+1)(k+2)} + \frac{2||x^{(0)} - \hat{x}||_\Gamma^2}{(k+1)(k+2)^{1-c}} + \frac{2(k+3)^{3-c}\,\mathsf{tr}\{\Gamma^{-1}\Sigma^2\}}{(3-c)(k+1)(k+2)}\right]. \tag{5.10}$$

Proof. Use the derivation in [29, Section 6.3] using

$$\sum_{i=0}^{k}(i+2)^{2-c} \leq \int_1^{k+1}(x+2)^{2-c}dx \leq \frac{(k+3)^{3-c}}{3-c}.$$

□

In Corollary 3, the choice $c^{(k)} = 1.5$ provides the rate

$$\mathbb{E}\left[\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x})\right] \leq \left(\max_{\substack{0\leq l\leq k\\ \forall j}} \sqrt{\frac{d_j + (l+2)^{1.5}\gamma_j}{d_j + (l+1)^{1.5}\gamma_j}}\right) \times O\left(\frac{2||x^{(0)} - \hat{x}||_D^2}{k^2} + \frac{2||x^{(0)} - \hat{x}||_\Gamma^2}{\sqrt{k}} + \frac{2\,\mathsf{tr}\{\Gamma^{-1}\Sigma^2\}}{1.5\sqrt{k}}\right), \tag{5.11}$$

achieving, on average, the optimal rate $O(1/\sqrt{k})$ in the presence of stochastic noise[2]. Corollary 3 shows that using $c \leq 1$ will suffer from error accumulation, using $1 < c < 1.5$ might provide faster initial convergence than $c = 1.5$ but does not achieve the optimal rate, and $c > 1.5$ will cause slower overall convergence rate than $c = 1.5$. So, we prefer $1 < c \leq 1.5$, for which we expect to eventually reach smaller cost function values than the choice $\Gamma^{(k)} = D$ (or $\Gamma = 0$) in (5.3), since we prevent the accumulation of error from OS methods by increasing the denominator $\Gamma^{(k)}$ as (5.7). In other words, the algorithm with $M > 1$ and (5.7) is slower

[2] Stochastic gradient algorithms (using only first-order information) cannot decrease the stochastic noise faster than the rate $O(1/\sqrt{k})$ [78], and the proposed relaxation scheme achieves the optimal rate [29].

than the choice of (5.3) initially, but eventually becomes faster and reaches the optimum on average.

In light of these trade-offs, we further consider using an increasing sequence $c^{(k)}$ that is upper-bounded by 1.5, hoping to provide fast overall convergence rate. We investigated the following choice:

$$c^{(k)} = 1 + 0.5\left(1 - \frac{\eta}{k+\eta}\right) \tag{5.12}$$

for $k \geq 0$ with a parameter $\eta > 0$. This choice of $c^{(k)}$ balances between fast initial acceleration and prevention of error accumulation. In other words, this increasing $c^{(k)}$ can provide faster initial convergence than a constant $c^{(k)} = 1.5$ (or $\eta = 0$), yet guarantees the optimal (asymptotic) rate $O(1/\sqrt{k})$ as the case $c^{(k)} = 1.5$, based on Corollary 2. We leave further optimization of $c^{(k)}$ as future work.

5.2.5 The choice of Γ

To choose Γ, we would like to minimize the upper bound on the right hand side of (5.10). For simplicity in designing Γ, we consider a fixed value of c in (5.10), we ignore the $\max_{0\leq l\leq k}\sqrt{\alpha^{(l)}}$ term, and we ignore the "+1" and "+2" factors added to k. The optimal Γ for k (sub)iterations is

$$\hat{\Gamma}_c(k) \triangleq \arg\min_{\Gamma\succ 0}\left\{\frac{||x^{(0)} - \hat{x}||_\Gamma^2}{k^{2-c}} + \frac{\mathsf{tr}\{\Gamma^{-1}\Sigma^2\}}{(3-c)k^{c-1}}\right\} = \mathsf{diag}\left\{\frac{\sigma_j}{\sqrt{3-c}\,\hat{u}_j}k^{1.5-c}\right\}, \tag{5.13}$$

where

$$\hat{u}_j \triangleq |x_j^{(0)} - \hat{x}_j|. \tag{5.14}$$

It is usually undesirable to have to select the (sub)iteration k before iterating. The choice $c = 1.5$ cancels out the parameter k in (5.13) leading to the k-independent choice:

$$\hat{\Gamma}_{1.5}(\cdot) = \mathsf{diag}\left\{\frac{\sigma_j}{\sqrt{1.5}\,\hat{u}_j}\right\}. \tag{5.15}$$

Since we prefer the choice of $c^{(k)}$ that eventually becomes 1.5 for the optimal rate, we focus on the choice $\hat{\Gamma}_{1.5}(\cdot)$ in (5.15). The next section describes practical approximations for the values of σ_j and $\hat{u}_j$ in (5.15).

5.3 Results

We investigate the convergence rate of the three proposed OS-momentum algorithms in Tables 5.1, 5.2 and 5.3, namely OS-mom1, OS-mom2 and OS-mom3 in this section, for PWLS reconstruction of simulated and real 3D CT scans. We implemented the proposed algorithm in C and ran on a Mac with two 2.26GHz quad-core Intel Xeon processors and a 16GB RAM using 16 threads.

We use an edge-preserving potential function $\psi_k(\cdot)$ in (2.13). For simulation data, the spatial weighting β_k was chosen to be

$$\beta_k \triangleq 50 \prod_{\forall j, c_{kj} \neq 0} \max\{\kappa_j, 0.01\kappa_{\max}\} \tag{5.16}$$

for uniform resolution properties [43], where $\kappa_j \triangleq \sqrt{\frac{\sum_{i=1}^{N_d} a_{ij} w_i}{\sum_{i=1}^{N_d} a_{ij}}}$ and $\kappa_{\max} \triangleq \max_j \kappa_j$. We emulated β_k of the GE product "Veo" for the patient 3D CT scans. We use the NU approach [62] in SQS methods for fast convergence. We investigated 12, 24 and 48 subsets for OS algorithms.

We first use simulated data to analyze the factors that affect the stability of the proposed OS-momentum algorithms, and further study the relaxation scheme for the algorithms. Then we verify the convergence speed of the proposed algorithm using real 3D CT scans. We computed the root mean squared difference (RMSD[3]) between the current and converged[4] image within the region-of-interest (ROI) versus computation time[5] for 30 iterations (n) to measure the convergence rate.[6]

5.3.1 Simulation data

We simulated $888 \times 64 \times 2934$ sinogram from a $1024 \times 1024 \times 154$ XCAT phantom [98] scanned in a helical geometry with pitch 1.0. We reconstructed $512 \times 512 \times 154$ images with an initial filtered back-projection (FBP) image $x^{(0)}$ using a (simple) single-slice rebinning method [88]. Fig. 5.1 shows that SQS-Nesterov's momentum methods without OS algorithms do not accelerate SQS much. OS algorithm itself can accelerate the SQS algorithm

[3] $\mathrm{RMSD}_{\mathrm{ROI}}(x^{(n)}) \triangleq ||x^{(n)}_{\mathrm{ROI}} - \hat{x}_{\mathrm{ROI}}||/\sqrt{N_{p,\mathrm{ROI}}}$ [HU], where $N_{p,\mathrm{ROI}}$ is the number of voxels within the ROI.

[4] We ran thousands of iterations of (convergent) SQS algorithm to generate (almost) converged images $\hat{x}$.

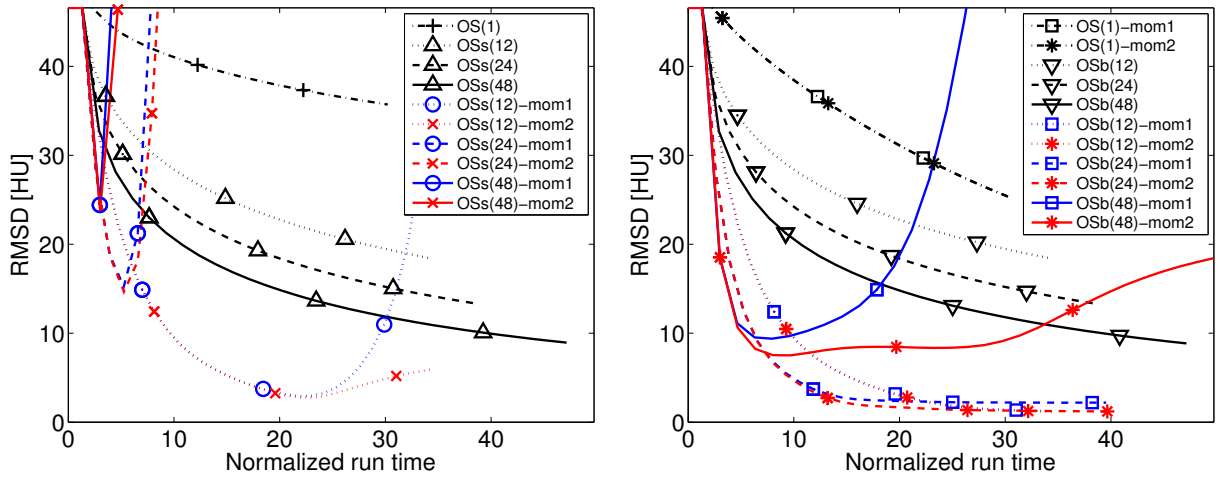
[5] We use the normalized run time that is a computation time divided by the run time required for one iteration of SQS (one subset version of OS-SQS) for each data set.

[6] Even though the convergence analysis in Section 5.2 is based on the cost function, we plot RMSD rather than the cost function because RMSD is more informative (see Section 4.2.3.4).

better than Nesterov's momentum. Our proposed OS-momentum algorithms rapidly decrease RMSD in early iterations (disregarding the diverging curves that we address shortly). However, the convergence behavior of OS-momentum algorithm depends on several factors such as the number of subsets, the order of subsets, and the type of momentum techniques. Thus, we discuss these in more detail based on the results in Fig. 5.1, and suggest ways to improve the stability of the algorithm while preserving the fast convergence rate.

5.3.1.1 The number of subsets

Intuitively, using more subsets will provide faster initial convergence but will increase instability due to errors between the full gradient and subset gradient. Also, performing many sub-iterations (m) can increase error accumulation per outer iteration (n). Fig. 5.1 confirms this behavior.



(a) Sequential order (OSs)

(b) Bit-reversal order (OSb)

Figure 5.1: Simulation data: convergence rate of OS algorithms (12, 24, 48 subsets) for 30 iterations with and without momentum for (a) sequential order and (b) bit-reversal order in Table 5.4. (The first iteration counts the precomputation of the denominator D in (2.19), and thus there are no changes during the first iteration.)

5.3.1.2 The ordering of subsets

Interestingly, the results of the proposed algorithms depend greatly on the subset ordering. Fig. 5.1 focuses on two deterministic orders: a sequential (OSs) order, and a bit-reversal (OSb) order [50] that selects each order-adjacent subsets to have their projection views to be far apart as described in Table 5.4. The ordering greatly affects the build-up of momentum in OS-momentum algorithms, whereas ordering is less important for ordinary OS methods as

seen in Fig. 5.1. The bit-reversal order provided much better stability in Fig. 5.1(b) compared to the results in Fig. 5.1(a). Apparently, the bit-reversal order can cancel out some gradient errors, because successive updates are likely to have opposite directions due to its subset ordering rule. In contrast, the sequential ordering has high correlation between the updates from two adjacent subsets, increasing error accumulation through momentum. Therefore, we recommend using the bit-reversal order. (Fig. 5.3(d) shows that random ordering (OSr) performed worse than the bit-reversal order.)

Sequential (OSs):	$(S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7), (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7), \cdots$
Bit-reversal (OSb):	$(S_0, S_4, S_2, S_6, S_1, S_5, S_3, S_7), (S_0, S_4, S_2, S_6, S_1, S_5, S_3, S_7), \cdots$
Random (OSr):	$S_6, S_7, S_1, S_7, S_5, S_0, S_2, S_4, S_7, S_3, S_2, S_3, S_0, S_4, S_1, S_6, \cdots$

Table 5.4: Examples of subset orderings: Two deterministic subset ordering (OSs, OSb) and one instance of random ordering (OSr) for OS methods with $M = 8$ subsets in a simple geometry with 24 projection views denoted as $(p_0, p_1, \cdots, p_{23})$, where those are reasonably grouped into the following 8 subsets: $S_0 = (p_0, p_8, p_{16})$, $S_1 = (p_1, p_9, p_{17})$, $\cdots$, $S_7 = (p_7, p_{15}, p_{23})$.

5.3.1.3 Type of momentum

Fig. 5.1 shows that combining OS with two of Nesterov's momentum techniques in Tables 5.1 and 5.2 (OS-mom1 and OS-mom2) resulted in different behaviors, whereas the one-subset versions of them behaved almost the same. Fig. 5.1 shows that the OS-mom2 algorithm is more stable than the OS-mom1 algorithm perhaps due to the different formulation of momentum or the fact that the momentum term $\{z^{(n+\frac{m}{M})}\}$ in Table 5.1 is not guaranteed to stay within the feasible set $\mathbb{R}_+^{N_p}$ unlike that in Table 5.2. Therefore, we recommend using the OS-mom2 algorithm in Table 5.2 for better stability.

Fig. 5.2 shows the initial image and the converged image and reconstructed images after 20 normalized run time (15 iterations) of conventional OS and our proposed OS-momentum algorithm with 24 subsets and the bit-reversal ordering. The OSb(24)-mom2 reconstructed image is very similar to the converged image after 15 iterations while that of conventional OS is still noticeably different. However, even the more stable OS-mom2 algorithm becomes unstable eventually for many subsets ($M > 24$) as seen in Fig. 5.1; the next subsection shows how relaxation improves stability.

5.3.1.4 The choice of Γ

Section 5.2.5 gives an optimized Γ in (5.7) that minimizes the right term of (5.10), *i.e.*, the gradient error term, on average. However, since the right term in (5.10) is a worst-case

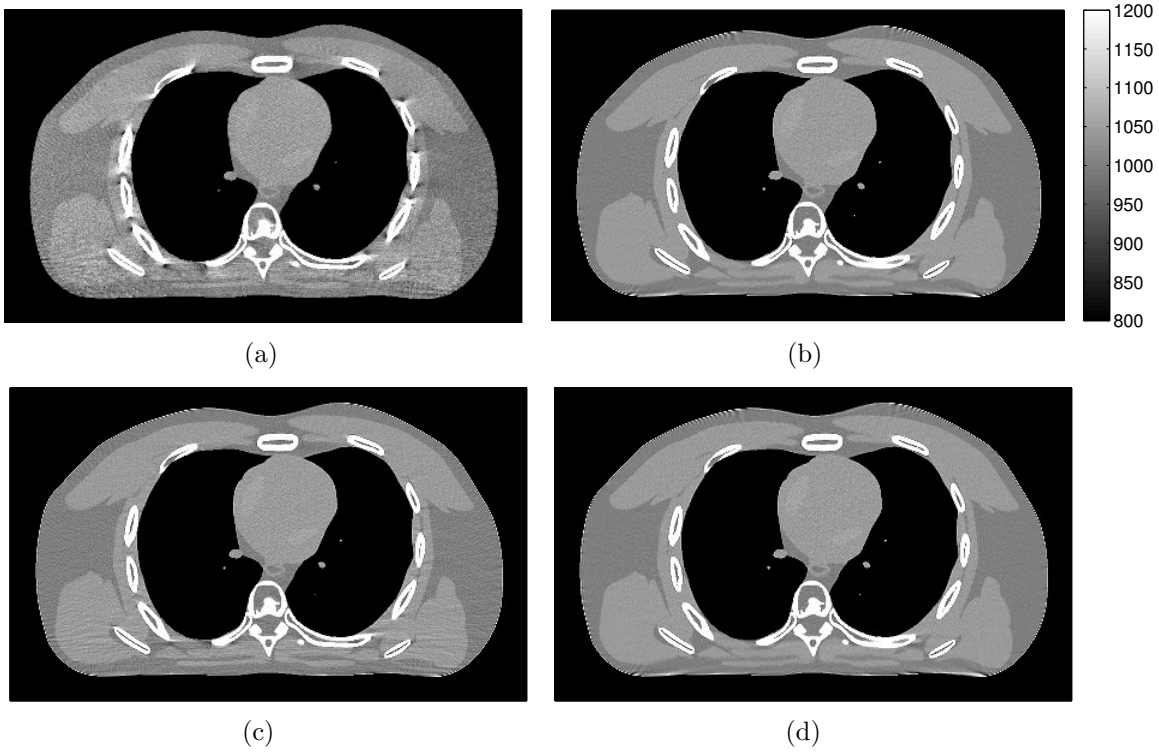


Figure 5.2: Simulation data: a transaxial plane of (a) an initial FBP image $x^{(0)}$, (b) a converged image $\hat{x}$, and two reconstructed images $x^{(15)}$ after 20 normalized run time (15 iterations) of (c) OSb(24) and (d) OSb(24)-mom2. (Images are cropped for better visualization.)

loose upper-bound, we can afford to use smaller Γ than $\hat{\Gamma}_{1.5}(\cdot)$ in (5.15). In addition, we may use even smaller Γ depending on the order of subsets. Specifically, the bit-reversal ordering (OSb) appears to accumulate less gradient error than other orderings, including random subset orders (OSr), so the choice (5.15) may be too conservative. Therefore, we investigated decreasing the matrix $\hat{\Gamma}_{1.5}(\cdot)$ (5.15) by a factor $\lambda \in (0,\ 1]$ as

$$\Gamma = \lambda \hat{\Gamma}_{1.5}(\cdot), \tag{5.17}$$

where we empirically investigate the parameter λ in Fig. 5.3.

The optimized $\hat{\Gamma}_{1.5}(\cdot)$ in (5.15) relies on unavailable parameters $\{\sigma_j\}$ and $\{\hat{u}_j\}$, so we provide a practical approach to estimate them, which we used in Fig. 5.3. The value $\sigma_j =$

$\max_{x\in\mathcal{B}} \tilde{\sigma}_j(x)$ in (5.2) is the upper bound of the values $\tilde{\sigma}_j(x)$ for all $x \in \mathcal{B}$, where

$$\tilde{\sigma}_j(x) \triangleq \sqrt{\mathbb{E}\left[\left(M\nabla_j \Psi_{S_k}(x) - \nabla_j \Psi(x)\right)^2\right]} = \sqrt{M \sum_{m=0}^{M-1} \left[A'_m W_m (A_m x - y_m)\right]_j^2 - \left[A' W (Ax - y)\right]_j^2}. \tag{5.18}$$

In practice, the sequences in Table 5.3 visit only a part of the set $\mathcal{B}$, so it would be preferable to compute $\tilde{\sigma}_j(x)$ within such part of $\mathcal{B}$ but even that is impractical. Instead, we use $\tilde{\sigma}_j(x^{(0)})$ as a practical approximation that is computationally efficient. This quantity measures the variance of the stochastic estimate of the gradient at the initial image $x^{(0)}$, mostly depending on the grouping of the subsets and the number of subsets.[7]

We compute value $\tilde{\sigma}_j(x^{(0)})$ in (5.18) simultaneously with the computation of D in (2.19) using modified projectors A and A' (see [62, Section III.F]) that handle two inputs and only slightly increase the computation rather than doubling the computation.

We further approximate $\hat{u}_j$ (5.14) by

$$\hat{u}_j \approx \zeta \bar{u} \tag{5.19}$$

for $\hat{\Gamma}_{1.5}(\cdot)$ in (5.15), where $\zeta > 0$ is an (unknown) constant, and a vector $\bar{u} \in \mathbb{R}_+^{N_p}$ is a normalized approximation of $\hat{u}_j$ that satisfies $\|\bar{u}_{\mathrm{ROI}}\|/\sqrt{N_{p,\mathrm{ROI}}} = 1$, which is computed by applying an edge-detector to the FBP image that is used for the initial $x^{(0)}$ [62, Section III.E].[8] In low-dose clinical CT, the RMSD within ROI between the initial FBP image and the converged image is about 30 [HU], *i.e.* $\|\hat{u}_{\mathrm{ROI}}\|/\sqrt{N_{p,\mathrm{ROI}}} \approx 30$ [HU], so we let $\zeta = 30$ [HU] in (5.19) as a reasonable choice in practice.[9] Then, our final practical choice of Γ becomes

$$\tilde{\Gamma} = \lambda \hat{\Gamma}_{1.5}(\cdot) \approx \lambda\, \mathsf{diag}\left\{ \frac{\tilde{\sigma}_j(x^{(0)})}{\sqrt{1.5}\, \zeta \bar{u}_j} \right\}. \tag{5.20}$$

In Fig. 5.3, we have investigated the parameter λ for various choices of the number and ordering of subsets. In all cases, $\lambda = 1$ is too conservative and yields very slow convergence. Smaller $\lambda \leq 0.1$ values lead to faster convergence, but it failed to stabilize the case of sequential ordering for $M > 24$. However, we found it reasonable to use $\lambda = 0.01$ for the bit-reversal orderings in the simulation data, while the choice $\lambda = 0.001$ was too small

[7] We found that $\sum_{j=1}^{N_p} \tilde{\sigma}_{48,j}(x^{(0)}) \approx 2.1 \sum_{j=1}^{N_p} \tilde{\sigma}_{24,j}(x^{(0)}) \approx 4.4 \sum_{j=1}^{N_p} \tilde{\sigma}_{12,j}(x^{(0)})$ for the simulation data, agreeing with the discussion in Section 5.3.1.1.

[8] We provide convergence results from using the oracle $\hat{u}$, compared to its approximate $\zeta\bar{u}$, in Section 5.3.3.

[9] This choice worked well in our experiments, but depend on the initial image, the cost function and the measurements, so improving the choice of ζ is future work.

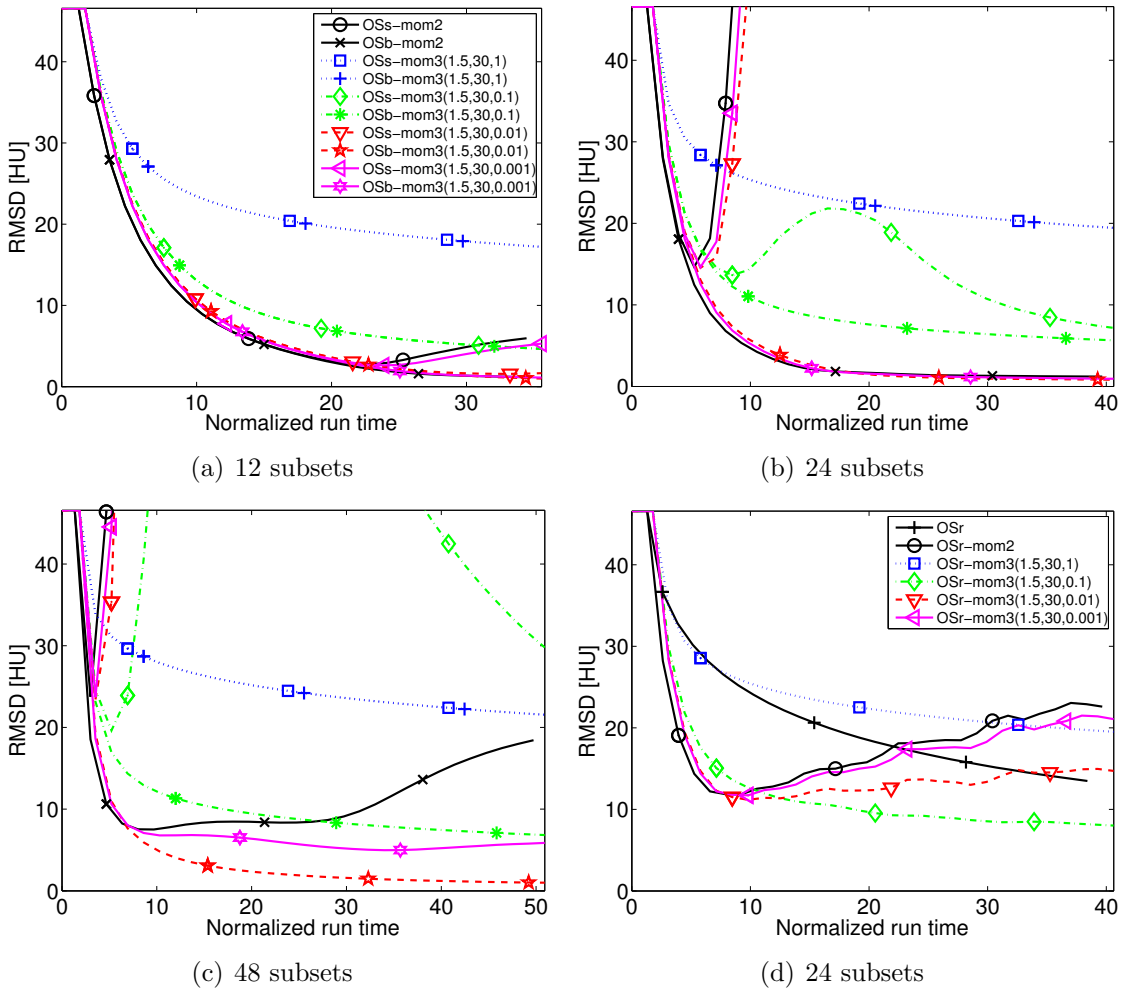


Figure 5.3: Simulation data: convergence rate for various choices of the parameter λ in relaxation scheme of OS-momentum algorithms (c, ζ, λ) for (a) 12, (b) 24, (c) 48 subsets with both sequential (OSs) and bit-reversal (OSb) subset orderings in Table 5.4 for 30 iterations. (The plot (b) and (c) share the legend of (a).) The averaged plot of five realizations of random subset ordering (OSr) is illustrated in (d) for 24 subsets.

to suppress the accumulation of error within 30 iterations for 48 subsets. Any value of $\lambda > 0$ here will eventually lead to stability as $\Gamma^{(k)}$ increases with $c^{(k)} = 1.5$, based on the convergence analysis (5.10). Particularly, OSs-mom3 algorithm with $\lambda = 0.1$ in Figs. 5.3(b) and 5.3(c) illustrates this stability property, where the RMSD curve recovers from the initial diverging behavior as the algorithm proceeds.

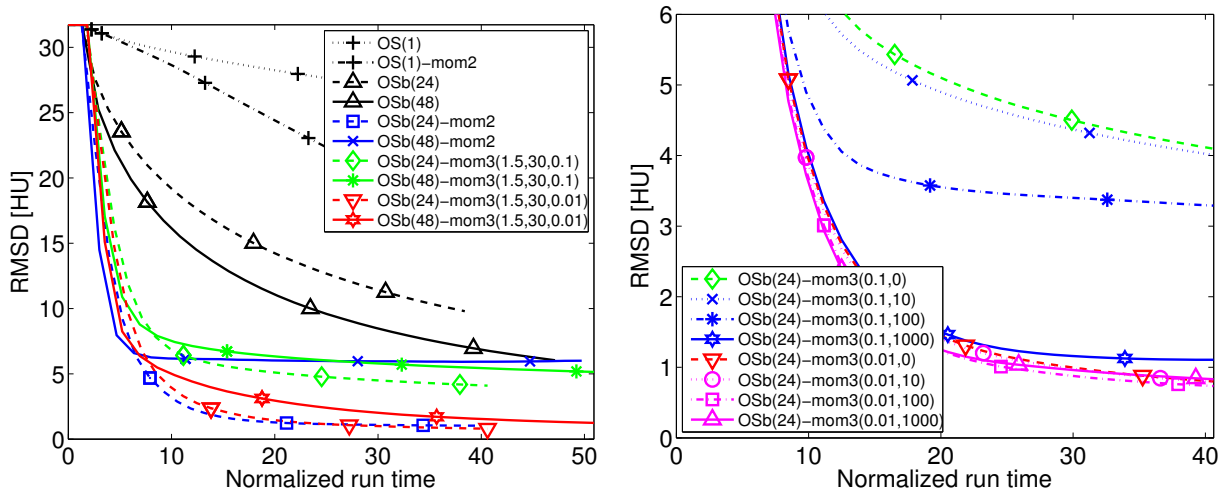
For Fig. 5.3(d), we executed five realizations of the random ordering and show the average of them for each curve. Here, we found that $\lambda = 0.01$ was too small to suppress the error within 30 iterations, and $\lambda = 0.1$ worked the best. Based on Fig. 5.3, we recommend using the bit-reversal order with $\lambda = 0.01$ rather than random ordering.

Figs. 5.1 and 5.3 are plotted with respect to the run time of each algorithm. Here using larger subsets increased the run time, about 0.01 normalized run time increment per each iteration (n) for increasing M by one in our implementation. The additional computation required for momentum methods was almost negligible, confirming that introducing momentum approach accelerates OS algorithm significantly in run time.

Overall, the simulation study demonstrated dramatic acceleration from combining OS algorithm and momentum approach. Next, we study the proposed OS-momentum algorithms on patient data, and verify that the parameters tuned with the simulation data work well for real CT scans.[10]

5.3.2 Shoulder region scan data

From a $888 \times 32 \times 7146$ sinogram measured in a helical geometry with pitch 0.5, we reconstructed a $512 \times 512 \times 109$ shoulder region image in Fig. 5.5. Fig. 5.4 shows the RMSD convergence curves for the bit-reversal subset ordering, where the results are similar to those for the simulation in Fig. 5.3 in terms of parameter selection. In Fig. 5.4(a), the parameter $\lambda = 0.01$ for both 24 and 48 subsets worked well providing overall fast convergence. Particularly for $M = 48$, the choice $\lambda = 0.01$ greatly reduced the gradient approximation error and converged faster than the un-relaxed OS-momentum algorithm.

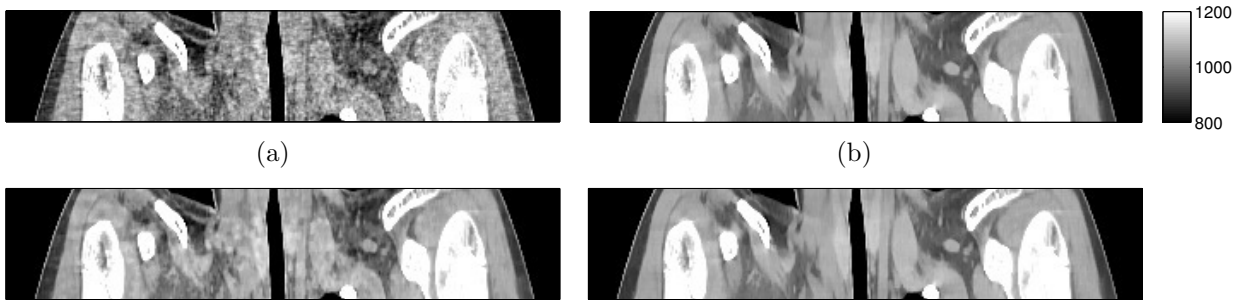


(a) (b)

Figure 5.4: Shoulder region scan data: convergence rate of OSb methods (24, 48 subsets) for 30 iterations with and without momentum for (a) several choices of (c, ζ, λ) with a fixed $c^{(k)} = c = 1.5$ and (b) the choices of (λ, η) for an increasing $c^{(k)}$ in (5.12) with 24 subsets and $\zeta = 30$ [HU].

[10] We present from another real CT scan in the supplementary material.

In Fig. 5.4(b), we further investigate the increasing $c^{(k)}$ (5.12) in (5.7) that starts from 1 and eventually becomes 1.5 with a tuning parameter η in (5.12). Larger η in (5.12) leads to a slowly increasing $c^{(k)}$, *i.e.* smaller $c^{(k)}$ values in early iterations (k), and thus, the results in Fig. 5.4(b) show better initial acceleration from using large η. Particularly, using large η for the choice $\lambda = 0.1$ showed a big acceleration, while that was less effective in the case $\lambda = 0.01$ due to small values of Γ (5.17) in (5.7).



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

Figure 5.5: Shoulder region scan data: a sagittal plane of (a) an initial FBP image $x^{(0)}$, (b) a converged image $\hat{x}$, and two reconstructed images $x^{(15)}$ after 20 normalized run time (15 iterations) from (c) OSb(24) and (d) OSb(24)-mom3 where $(c, \zeta, \lambda) = (1.5, 30, 0.01)$.

Fig. 5.5 shows the initial FBP image, converged, and reconstructed images from conventional OS and the proposed OS-momentum with relaxation. Visually, the reconstructed image from the proposed algorithm is almost identical to the converged image after 15 iterations.

5.3.3 Abdominal region scan

We reconstructed a $600 \times 600 \times 222$ abdominal region image (in Fig. 5.6) from $888 \times 64 \times 3611$ sinogram data measured in a helical geometry with pitch 1.0. We measured the RMSD of the proposed OS-momentum algorithms for 24 and 48 subsets using bit-reversal ordering (OSb) in OS methods. The convergence results in Fig. 5.7(a) are similar to that of previous two data sets, where $\lambda = 0.01$ is more effective than $\lambda = 0.1$ for stabilizing OS-momentum. However, the case $M = 48$ accumulated more gradient error than two other data sets, particularly the un-relaxed ($\lambda = 0$) OS-momentum for $M = 48$ become very unstable. So the choice $\lambda = 0.01$ was not large enough to suppress the large accumulating error in first 10 iterations than $\lambda = 0.1$.

Fig. 5.7(b) shows the results using the oracle $\hat{u}_j = |x_j^{(0)} - \hat{x}_j|$ in (5.14) for 48 subsets, compared to those using the approximate $\zeta\bar{u}$ (5.19) of $\hat{u}$. The oracle parameter $\hat{u}$ worked

well for both $\lambda = 0.1$ and 0.01, indicating that the convergence rate depends less on λ when the voxel-dependent factor $\zeta \bar{u}_j$ better approximates $\hat{u}_j$, and we leave this as future work.

Fig. 5.6 shows the initial FBP image, converged image, and the reconstructed image at 15th iteration from the proposed algorithm. The reconstructed image is very close to the converged image after 15 iterations, largely removing the streak artifacts in FBP.

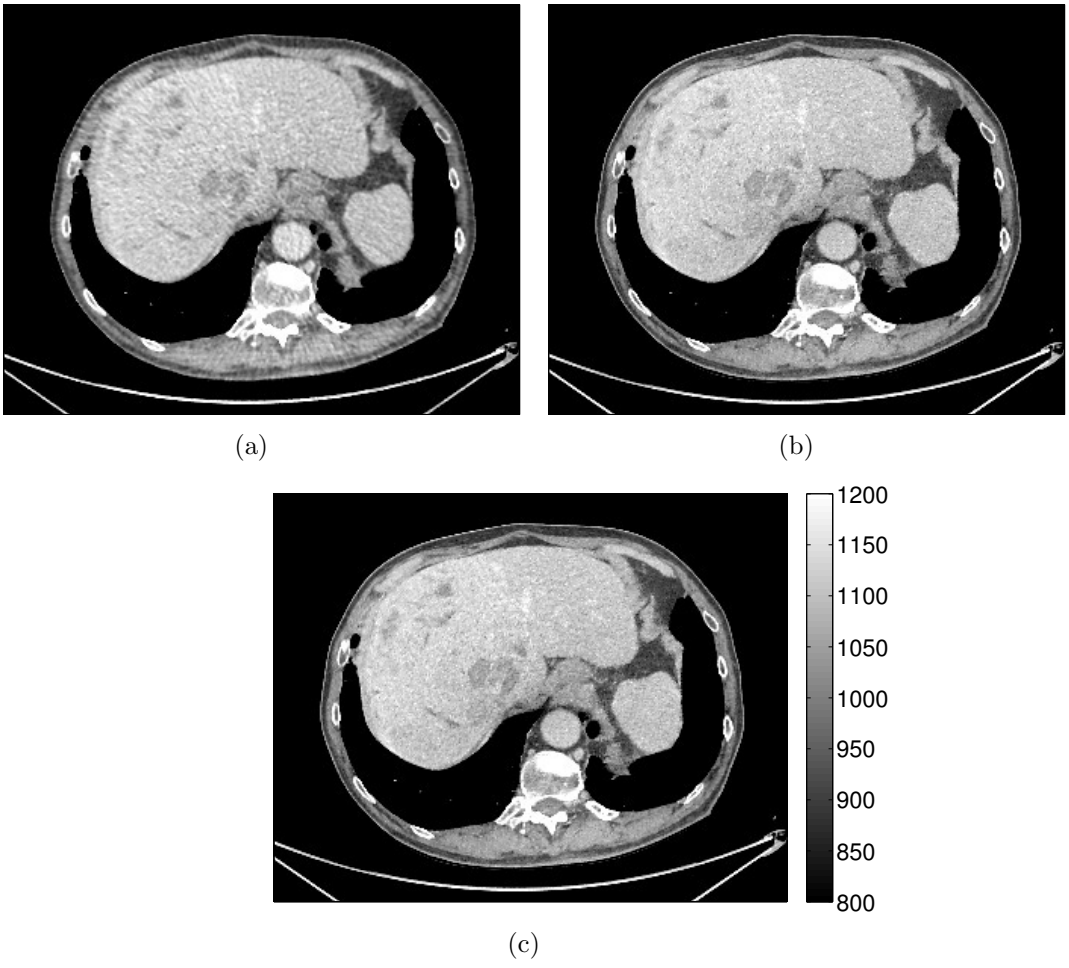
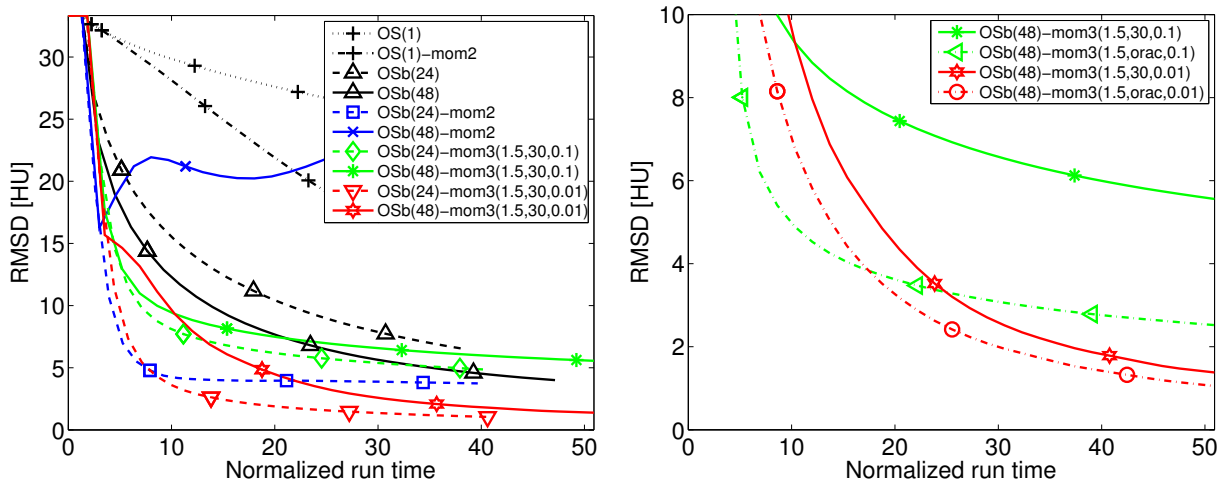


Figure 5.6: Abdominal region scan: a transaxial plane of (a) an initial FBP image $x^{(0)}$, (b) a converged image $\hat{x}$, and (c) an image $x^{(15)}$ after 20 normalized run time (15 iterations) of OSb(24)-mom3 where $(c, \zeta, \lambda) = (1.5, 30, 0.01)$. (Images are cropped for better visualization.)

5.4 Conclusion and Discussion

We introduced the combination of OS-SQS and Nesterov's momentum techniques in tomography problems. We quantified the accelerated convergence of the proposed algorithms using simulated and patient 3D CT scans. The initial combination could lack stability for



(a) (b)

Figure 5.7: Abdominal region scan: convergence rate of OSb methods (24, 48 subsets) for 30 iterations with and without momentum for several choices of (c, ζ, λ) with (a) the choice $\zeta\bar{u}(\approx \hat{u})$ in (5.19) with $\zeta = 30$ [HU] and (b) the oracle choice $\hat{u}$ (5.14) for 48 subsets.

large numbers of subsets, depending on the subset ordering and type of momentum. So, we adapted a diminishing step size approach to stabilize the proposed algorithm while preserving fast convergence.

We have focused on PWLS cost function in this chapter, but the proposed algorithms can be applied to any (smooth or nonsmooth) convex cost function, including the Poisson. We are further interested in studying the proof of convergence of the OS-momentum algorithm for a (bit-reversal) "deterministic" order.

The accumulating error of the proposed algorithms in Section 5.2 is hard to measure due to the computational complexity, and thus optimizing the relaxation parameters for an increasing $\Gamma^{(k)}$ in (5.7) remains an open issue. In our experiments, we observed that the updated images within subiterations of OS-momentum algorithms form a cycle looping around the optimal image $\hat{x}$ with increasing radius when they reach a diverging phase. (The plain OS eventually reaches a fixed limit-cycle.) One could consider this behavior as an indication of instability that could lead to adaptively decreasing the step size, *i.e.*, increasing $\Gamma^{(k)}$. Alternatively, one could discard the current momentum and restart the build-up of the momentum as in [72, 90]. Such refinements could make OS-momentum a practical approach for low-dose CT.

CHAPTER VI

Optimized momentum approaches

We introduce new optimized first-order methods for smooth unconstrained convex minimization. Drori and Teboulle [31] recently described a numerical method for computing the N-iteration optimal step coefficients in a class of first-order algorithms that includes a gradient method, a heavy-ball method [93], and Nesterov's fast gradient methods [80, 83]. However, the numerical method in [31] is computationally expensive for large N, and the corresponding numerically optimized first-order algorithm in [31] requires impractical memory for large-scale optimization problems and $O(N^2)$ computation for N iterations. In this chapter, we propose optimized first-order algorithms that achieve a convergence bound that is two times faster than Nesterov's fast gradient methods; our bound is found analytically and refines the numerical bound in [31]. Furthermore, we show that the proposed optimized first-order methods have efficient recursive forms that are remarkably similar to Nesterov's fast gradient methods and require $O(N)$ computations for N iterations.

6.1 Introduction

First-order algorithms are used widely to solve large-scale optimization problems in various fields such as signal and image processing, machine learning, communications and many other areas. The computational cost per iteration of first-order algorithms is mildly dependent on the dimension of the problem, yielding computational efficiency. Particularly, Nesterov's fast gradient methods [80, 83] have been celebrated in various applications for their fast convergence rates and efficient recursive implementation. This chapter proposes first-order algorithms (OGM1 and OGM2 in Section 6.6) that are twice as fast (in terms of worst-case bounds) as Nesterov's fast gradient methods for smooth unconstrained convex minimization yet require remarkably similar computation per iteration.

We consider finding a minimizer over $\mathbb{R}^d$ of a cost function f belonging to a set $\mathcal{F}_L(\mathbb{R}^d)$ of smooth convex functions with L-Lipschitz continuous gradient. The class of first-order

algorithms of interest generates a sequence of points $\{\boldsymbol{x}_i \in \mathbb{R}^d \ : \ i = 0, \cdots, N\}$ using the following scheme:

Algorithm Class FO

Input: $f \in \mathcal{F}_L(\mathbb{R}^d)$, $\boldsymbol{x}_0 \in \mathbb{R}^d$.

For $i = 0, \cdots, N-1$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{1}{L} \sum_{k=0}^{i} h_{i+1,k} f'(\boldsymbol{x}_k). \tag{6.1}$$

The update step at the ith iterate $\boldsymbol{x}_i$ uses a linear combination of previous and current gradients $\{f'(\boldsymbol{x}_0), \cdots, f'(\boldsymbol{x}_i)\}$. The coefficients $\{h_{i,k}\}_{0 \le k < i \le N}$ determine the step size and are selected prior to iterating (non-adaptive). Designing these coefficients appropriately is the key to establishing fast convergence. The algorithm class FO includes a gradient method, a heavy-ball method [93], Nesterov's fast gradient methods [80,83], and our proposed optimized first-order methods.

Evaluating the convergence rate of such first-order algorithms is essential. Recently, Drori and Teboulle (hereafter "DT") [31] considered the Performance Estimation Problem (PEP) approach to bounding the decrease of a cost function f. For given coefficients $\boldsymbol{h} = \{h_{i,k}\}_{0 \le k < i \le N}$, a given number of iterations $N \ge 1$ and a given upper bound $R > 0$ on the distance between an initial point $\boldsymbol{x}_0$ and an optimal point $\boldsymbol{x}_* \in X_*(f) \triangleq \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x})$, the worst-case performance bound of a first-order method over all smooth convex functions $f \in \mathcal{F}_L(\mathbb{R}^d)$ is the solution of the following constrained optimization problem [31]:

$$\begin{aligned} \mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R) \triangleq \max_{f \in \mathcal{F}_L(\mathbb{R}^d)} \max_{\substack{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d, \\ \boldsymbol{x}_* \in X_*(f)}} & f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \\ \text{s.t. } \boldsymbol{x}_{i+1} &= \boldsymbol{x}_i - \frac{1}{L} \sum_{k=0}^{i} h_{i+1,k} f'(\boldsymbol{x}_k), \quad i = 0, \cdots, N-1, \\ ||\boldsymbol{x}_0 - \boldsymbol{x}_*|| &\le R. \end{aligned} \tag{P}$$

As reviewed in Section 6.4.1, DT [31] used relaxations to simplify the intractable problem (P) to a solvable form.

Nesterov's fast gradient methods [80, 83] achieve the optimal rate of decrease $O\left(\frac{1}{N^2}\right)$ of $\mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R)$ for minimizing a smooth convex function f [81]. Seeking first-order algorithms that converge faster (in terms of the constant factor) than Nesterov's fast gradient methods, DT [31] proposed using a (relaxed) PEP approach to optimize the choice of $\boldsymbol{h}$ in class FO by minimizing a (relaxed) worst-case bound $\mathcal{B}(\boldsymbol{h}, N, L, R)$ at the Nth iteration with respect to

$\boldsymbol{h}$. In [31], the optimized $\boldsymbol{h}$ factors were computed numerically, and were found to yield faster convergence than Nesterov's methods. However, numerical optimization of $\boldsymbol{h}$ in [31] becomes expensive for large N. In addition, the general class FO, including the algorithm in [31], requires $O(N^2)$ computations for N iterations and requires $O(Nd)$ memory for storing all gradients $\{f'(\boldsymbol{x}_i) \in \mathbb{R}^d \ : \ i = 0, \cdots, N-1\}$, which is impractical for large-scale problems.

This chapter proposes optimized first-order algorithms that are twice as fast (in terms of worst-case bound) as Nesterov's fast gradient methods, inspired by [31]. We develop remarkably efficient recursive formulations of the optimized first-order algorithms that resemble those of Nesterov's fast gradient methods, requiring $O(N)$ computations and $O(d)$ memory.

Section 6.2 reviews the smooth convex minimization problem and introduces the approach to optimizing $\boldsymbol{h}$ used here and in [31]. Section 6.3 illustrates well-known examples in class FO. Section 6.4 reviews DT's PEP approach and uses the PEP approach to derive a new convergence bound for the *primary* variables in Nesterov's fast gradient methods. Section 6.5 reviews DT's analysis of a relaxed convergence bound for a first-order method, and derives a new equivalent analytical bound and step size coefficients $\boldsymbol{h}$ that optimize that bound. Section 6.6 investigates efficient formulations of the proposed first-order methods that achieve the relaxed convergence bound. Section 6.7 offers conclusions.

6.2 Problem and approach

6.2.1 Smooth convex minimization problem

We consider first-order algorithms for solving the following minimization problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \ f(\boldsymbol{x}), \tag{M}$$

where the following two conditions are assumed:

- $f \ : \ \mathbb{R}^d \to \mathbb{R}$ is a convex function of the type $C_L^{1,1}(\mathbb{R}^d)$, *i.e.*, continuously differentiable with Lipschitz continuous gradient:

  $$||f'(\boldsymbol{x}) - f'(\boldsymbol{y})|| \leq L||\boldsymbol{x} - \boldsymbol{y}||, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d,$$

  where $L > 0$ is the Lipschitz constant.

- The optimal set $X_*(f) = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x})$ is nonempty, *i.e.*, the problem (M) is solvable.

We focus on measuring the "inaccuracy" $f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*)$ after N iterations to quantify the worst-case performance of any given first-order algorithm.

6.2.2 Optimizing the step coefficients of first-order algorithms

In search of the best-performing first-order methods, DT [31] proposed to optimize $\boldsymbol{h} = \{h_{i,k}\}_{0\le k<i\le N}$ in Algorithm FO by minimizing the worst-case bound of $f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*)$ for a given number of iterations $N \ge 1$ and initial distance $R > 0$, by adding $\arg\min_{\boldsymbol{h}}$ to problem (P) as follows:

$$\hat{\boldsymbol{h}}_{\mathrm{P}} \triangleq \underset{\boldsymbol{h}\in\mathbb{R}^{N(N+1)/2}}{\arg\min}\ \mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R). \tag{HP}$$

Note that $\hat{\boldsymbol{h}}_{\mathrm{P}}$ is independent[1] of both L and R. Solving problem (HP) would give the step coefficients of the optimal first-order algorithm achieving the best worst-case convergence bound. DT [31] relaxed[2] problem (HP) to a tractable form, as reviewed in Sections 6.4.1 and 6.5.1. After these simplifications, the resulting solution was computed numerically using a semidefinite programming (SDP) that remains computationally expensive for large N [31]. In addition, the corresponding numerically optimized first-order algorithm was impractical for large-scale problems, requiring a linear combination of previous and current gradients $\{f'(\boldsymbol{x}_0), \cdots, f'(\boldsymbol{x}_i)\}$ at the $(i+1)$-th iteration.[3]

To make DT's work [31] practical, we directly derive the "analytical" solution for $\boldsymbol{h}$ in a relaxed version of the problem (HP), circumventing the numerical approach in [31]. Interestingly, the analytical solution of the relaxed version of (HP) satisfies a convenient recursion, so we provide practical optimized algorithms similar to efficient Nesterov's fast gradient methods.

Next, we provide examples of Algorithm FO such as a gradient method, a heavy-ball method [93], and Nesterov's fast gradient methods [80, 83].

---

[1] Using the substitutions $\boldsymbol{x}' = \frac{1}{R}\boldsymbol{x}$ and $\breve{f}(\boldsymbol{x}') = \frac{1}{LR^2} f(R\boldsymbol{x}') \in \mathcal{F}_1(\mathbb{R}^d)$ in problem (P), we get $\mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R) = LR^2\mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, 1, 1)$. This leads to $\hat{\boldsymbol{h}}_{\mathrm{P}} = \arg\min_{\boldsymbol{h}} \mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R) = \arg\min_{\boldsymbol{h}} \mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, 1, 1)$.

[2] Using the term 'best' or 'optimal' here for [31] may be too strong, since [31] relaxed (HP) to a solvable form. We also use these relaxations, so we use the term "optimized" for our proposed algorithms.

[3] If coefficients $\boldsymbol{h}$ in Algorithm FO have a special recursive form, it is possible to find an equivalent efficient form, as discussed in Sections 6.3 and 6.6.

6.3 Examples of first-order algorithms

This section reviews examples of Algorithm FO as discussed in [31, Section 4.1]. We further show the equivalence[4] of two of Nesterov's fast gradient methods in smooth unconstrained convex minimization. The analysis techniques used here will be important in Section 6.6.

6.3.1 Gradient method

The conventional gradient method (GM) is one simple case of Algorithm FO when

$$h_{i+1,k} = \begin{cases} 0, & k = 0, \cdots, i-1, \\ 1, & k = i, \end{cases}$$

for $i = 0, \cdots, N-1$. This GM method is widely known to satisfy [10, 81]:

$$f(\boldsymbol{x}_n) - f(\boldsymbol{x}_*) \leq \frac{L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{2n}, \quad \forall \boldsymbol{x}_* \in X_*(f) \tag{6.2}$$

for $n \geq 1$. Recently, DT, using a specific PEP approach[5], found the tightest worst-case bound for a GM method [31, Theorems 1 and 2]:

$$f(\boldsymbol{x}_n) - f(\boldsymbol{x}_*) \leq \frac{L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{4n+2}, \quad \forall \boldsymbol{x}_* \in X_*(f) \tag{6.3}$$

for $n \geq 1$, which is twice better than the conventional bound (6.2).

6.3.2 Heavy-ball method

Another first-order method is a heavy-ball method (HBM) [93] that reuses the last update $\boldsymbol{x}_i - \boldsymbol{x}_{i-1}$ for acceleration with minimal extra computation, where the parameters $0 < \alpha < 2(1+\beta)$ and $0 \leq \beta < 1$ are used in [93].

[4] The equivalence of two of Nesterov's fast gradient methods for smooth unconstrained convex minimization was previously mentioned without details in [105].

[5] The PEP approach for GM is a special case in [31] where DT found explicitly the analytical convergence bound (6.3). In addition, DT specified a worst-case function in the class of convex $C_L^{1,1}$ that attains the bound (6.3), so (6.3) is the tightest for GM.

Algorithm HBM

Input: $f \in C_L^{1,1}(\mathbb{R}^d)$ convex, $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\boldsymbol{x}_{-1} = \boldsymbol{x}_0$.

For $i = 0, \cdots, N-1$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{\alpha}{L} f'(\boldsymbol{x}_i) + \beta(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$$

Algorithm HBM is an example of Algorithm Class FO by defining

$$h_{i+1,k} = \alpha\beta^{i-k}, \quad k = 0, \cdots, i,$$

for $i = 0, \cdots, N-1$ [31]. The convergence bound of HBM is unknown on the class of convex functions in $C_L^{1,1}$, and the PEP approach provided a numerical convergence bound [31, Fig. 1 and Table 1] that was found to be faster than that of the GM but slower than that of Nesterov's fast gradient methods.

Next, we review two of Nesterov's celebrated fast gradient methods [80,83]. These achieve the optimal convergence rate $O\left(\frac{1}{N^2}\right)$ for decreasing the function f [81], while requiring minimal additional computation compared to GM and HBM.

6.3.3 Nesterov's fast gradient method 1

Nesterov's first fast gradient method is called FGM1 [80]:

Algorithm FGM1

Input: $f \in C_L^{1,1}(\mathbb{R}^d)$ convex, $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\boldsymbol{y}_0 = \boldsymbol{x}_0$, $t_0 = 1$.

For $i = 0, \cdots, N-1$

$$\begin{aligned} \boldsymbol{y}_{i+1} &= \boldsymbol{x}_i - \frac{1}{L} f'(\boldsymbol{x}_i) \\ t_{i+1} &= \frac{1 + \sqrt{1 + 4t_i^2}}{2} \\ \boldsymbol{x}_{i+1} &= \boldsymbol{y}_{i+1} + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{y}_{i+1} - \boldsymbol{y}_i). \end{aligned} \tag{6.4}$$

Note that t_i in (6.4) satisfies the following relationships used frequently in later derivations:

$$t_{i+1}^2 - t_{i+1} - t_i^2 = 0, \quad t_i^2 = \sum_{k=0}^{i} t_k, \quad \text{and} \quad t_i \geq \frac{i+2}{2}, \quad i = 0, 1, \cdots. \tag{6.5}$$

Algorithm FGM1 is in Algorithm Class FO [31, Proposition 2] with:

$$\bar{h}_{i+1,k} = \begin{cases} \frac{t_i-1}{t_{i+1}}\bar{h}_{i,k}, & k = 0, \cdots, i-2, \\ \frac{t_i-1}{t_{i+1}}(\bar{h}_{i,i-1} - 1), & k = i-1, \\ 1 + \frac{t_i-1}{t_{i+1}}, & k = i, \end{cases} \tag{6.6}$$

for $i = 0, \cdots, N-1$. Note that Algorithm FO with (6.6) is impractical as written for large-scale optimization problems, whereas the mathematically equivalent version FGM1 is far more useful practically due to its efficient recursive form.

The sequence $\{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{N-1}, \boldsymbol{y}_N\}$ of FGM1 can be also written in class FO [31, Proposition 2], and the sequence $\{\boldsymbol{y}_0, \cdots, \boldsymbol{y}_N\}$ is known to achieve the rate $O\left(\frac{1}{N^2}\right)$ for decreasing f [10, 80]. DT conjectured that the *primary*[6] sequence $\{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N\}$ of FGM1 also achieves the same $O\left(\frac{1}{N^2}\right)$ rate based on the numerical results using the PEP approach [31, Conjecture 2]; our Section 6.4.2 verifies the conjecture by providing an analytical bound using the PEP approach.

6.3.4 Nesterov's fast gradient method 2

In [83], Nesterov proposed another fast gradient method that has a different form than FGM1 and that used a choice of t_i factors different from (6.4). Here, we use (6.4) because it leads to faster convergence than the factors used in [83]. The algorithm in [83] then becomes FGM2 shown below.

Algorithm FGM2

Input: $f \in C_L^{1,1}(\mathbb{R}^d)$ convex, $\boldsymbol{x}_0 \in \mathbb{R}^d$, $t_0 = 1$.

For $i = 0, \cdots, N-1$

$$\boldsymbol{y}_{i+1} = \boldsymbol{x}_i - \frac{1}{L} f'(\boldsymbol{x}_i)$$

$$\boldsymbol{z}_{i+1} = \boldsymbol{x}_0 - \frac{1}{L} \sum_{k=0}^{i} t_k f'(\boldsymbol{x}_k)$$

$$t_{i+1} = \frac{1 + \sqrt{1 + 4t_i^2}}{2}$$

$$\boldsymbol{x}_{i+1} = \left(1 - \frac{1}{t_{i+1}}\right) \boldsymbol{y}_{i+1} + \frac{1}{t_{i+1}} \boldsymbol{z}_{i+1}$$

[6] The sequence $\{\boldsymbol{x}_i\}$ of FGM1 was originally introduced to be auxiliary in FGM1 with no known convergence bound. In view of class FO, $\{\boldsymbol{x}_i\}$ is the *primary* sequence of FGM1 and we discuss its convergence bound in Section 6.4.2.

Similar to FGM1, the following proposition shows that FGM2 is in class FO with

$$\bar{h}_{i+1,k} = \begin{cases} \frac{1}{t_{i+1}} \left(t_k - \sum_{j=k+1}^{i} \bar{h}_{j,k} \right), & k = 0, \cdots, i-1, \\ 1 + \frac{t_i - 1}{t_{i+1}}, & k = i, \end{cases} \tag{6.7}$$

for $i = 0, \cdots, N-1$ with t_i in (6.4).

Proposition 1. *The points $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with* (6.7) *are identical to the respective points generated by Algorithm FGM2.*

Proof. We use induction to show that the sequence $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with (6.7) is identical to the respective points generated by Algorithm FGM2. For clarity, we use the notation $\boldsymbol{x}'_0, \cdots, \boldsymbol{x}'_N$ for Algorithm FO.

Clearly $\boldsymbol{x}'_0 = \boldsymbol{x}_0$. To prove the equivalence for $i = 1$:

$$\begin{aligned} \boldsymbol{x}'_1 &= \boldsymbol{x}'_0 - \frac{1}{L} \bar{h}_{1,0} f'(\boldsymbol{x}'_0) = \boldsymbol{x}_0 - \frac{1}{L} \left(1 + \frac{t_0 - 1}{t_1} \right) f'(\boldsymbol{x}_0) \\ &= \left(1 - \frac{1}{t_1} + \frac{1}{t_1} \right) \left(\boldsymbol{x}_0 - \frac{1}{L} f'(\boldsymbol{x}_0) \right) = \left(1 - \frac{1}{t_1} \right) \boldsymbol{y}_1 + \frac{1}{t_1} \boldsymbol{z}_1 = \boldsymbol{x}_1. \end{aligned}$$

Assuming $\boldsymbol{x}'_i = \boldsymbol{x}_i$ for $i = 0, \cdots, n$, we then have

$$\begin{aligned} \boldsymbol{x}'_{n+1} =& \boldsymbol{x}'_n - \frac{1}{L} \bar{h}_{n+1,n} f'(\boldsymbol{x}'_n) - \frac{1}{L} \sum_{k=0}^{n-1} \bar{h}_{n+1,k} f'(\boldsymbol{x}'_k) \\ =& \boldsymbol{x}_n - \frac{1}{L} \left(1 + \frac{t_n - 1}{t_{n+1}} \right) f'(\boldsymbol{x}_n) - \frac{1}{L} \sum_{k=0}^{n-1} \frac{1}{t_{n+1}} \left(t_k - \sum_{j=k+1}^{n} \bar{h}_{j,k} \right) f'(\boldsymbol{x}_k) \\ =& \left(1 - \frac{1}{t_{n+1}} \right) \left(\boldsymbol{x}_n - \frac{1}{L} f'(\boldsymbol{x}_n) \right) + \frac{1}{t_{n+1}} \left(\boldsymbol{x}_n + \frac{1}{L} \sum_{k=0}^{n-1} \sum_{j=k+1}^{n} \bar{h}_{j,k} f'(\boldsymbol{x}_k) - \frac{1}{L} \sum_{k=0}^{n} t_k f'(\boldsymbol{x}_k) \right) \\ =& \left(1 - \frac{1}{t_{n+1}} \right) \boldsymbol{y}_{n+1} + \frac{1}{t_{n+1}} \left(\boldsymbol{x}_n + \frac{1}{L} \sum_{j=1}^{n} \sum_{k=0}^{j-1} \bar{h}_{j,k} f'(\boldsymbol{x}_k) - \frac{1}{L} \sum_{k=0}^{n} t_k f'(\boldsymbol{x}_k) \right) \\ =& \left(1 - \frac{1}{t_{n+1}} \right) \boldsymbol{y}_{n+1} + \frac{1}{t_{n+1}} \left(\boldsymbol{x}_0 - \frac{1}{L} \sum_{k=0}^{n} t_k f'(\boldsymbol{x}_k) \right) \\ =& \left(1 - \frac{1}{t_{n+1}} \right) \boldsymbol{y}_{n+1} + \frac{1}{t_{n+1}} \boldsymbol{z}_{n+1} = \boldsymbol{x}_{n+1}. \end{aligned}$$

The fifth equality uses the telescoping sum $\boldsymbol{x}_n = \boldsymbol{x}_0 + \sum_{j=1}^{n} (\boldsymbol{x}_j - \boldsymbol{x}_{j-1})$ and (6.1) in Algorithm FO. □

We show next the equivalence of Nesterov's two algorithms FGM1 and FGM2 for smooth unconstrained convex minimization using (6.6) and (6.7).

Proposition 2. *The points $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FGM2 are identical to the respective points generated by Algorithm FGM1.*

Proof. We prove the statement by showing the equivalence of (6.6) and (6.7). We use the notation $\bar{h}'_{i,k}$ for the coefficients (6.7) of Algorithm FGM2 to distinguish from those of Algorithm FGM1.

It is obvious that $\bar{h}'_{i+1,i} = \bar{h}_{i+1,i},\ i = 0, \cdots, N-1$, and we can easily prove that

$$\begin{aligned}\bar{h}'_{i+1,i-1} &= \frac{1}{t_{i+1}}\left(t_{i-1} - \bar{h}'_{i,i-1}\right) = \frac{1}{t_{i+1}}\left(t_{i-1} - \left(1 + \frac{t_{i-1}-1}{t_i}\right)\right) \\ &= \frac{(t_i - 1)(t_{i-1}-1)}{t_i t_{i+1}} = \frac{t_i - 1}{t_{i+1}}\left(\bar{h}_{i,i-1} - 1\right) = \bar{h}_{i+1,i-1}\end{aligned}$$

for $i = 0, \cdots, N-1$.

We next use induction by assuming $\bar{h}'_{i+1,k} = \bar{h}_{i+1,k}$ for $i = 0, \cdots, n-1,\ k = 0, \cdots, i$. We then have

$$\begin{aligned}\bar{h}'_{n+1,k} &= \frac{1}{t_{n+1}}\left(t_k - \sum_{j=k+1}^{n} \bar{h}'_{j,k}\right) = \frac{1}{t_{n+1}}\left(t_k - \sum_{j=k+1}^{n-1} \bar{h}'_{j,k} - \bar{h}'_{n,k}\right) \\ &= \frac{1}{t_{n+1}}\left(t_k - \left(t_k - t_n \bar{h}'_{n,k}\right) - \bar{h}'_{n,k}\right) = \frac{t_n - 1}{t_{n+1}}\bar{h}'_{n,k} = \frac{t_n - 1}{t_{n+1}}\bar{h}_{n,k} = \bar{h}_{n+1,k}\end{aligned}$$

for $k = 0, \cdots, n-2$. □

Algorithms FGM1 and FGM2 generate the same sequences $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_i\}$, and the sequence $\{\boldsymbol{y}_i\}$ is known to satisfy [10, 80, 83]:

$$f(\boldsymbol{y}_n) - f(\boldsymbol{x}_*) \leq \frac{2L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{(n+1)^2}, \quad \forall \boldsymbol{x}_* \in X_*(f) \tag{6.8}$$

for $n \geq 1$, which was the previously best known analytical bound of first-order methods for smooth unconstrained convex minimization; DT's PEP approach provides a tighter 'numerical' bound for the sequences $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_i\}$ compared to the analytical bound (6.8) [31, Table 1]. Using the PEP approach, Section 6.4.2 provides a previously unknown 'analytical' bound for the sequence $\{\boldsymbol{x}_i\}$ of FGM1 and FGM2.

Nesterov described a convex function $f \in C_L^{1,1}(\mathbb{R}^d)$ for which any first-order algorithm

generating the sequence $\{\boldsymbol{x}_i\}$ in a class of Algorithm FO satisfies [81, Theorem 2.1.7]:

$$\frac{3L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{32(n+1)^2} \leq f(\boldsymbol{x}_n) - f(\boldsymbol{x}_*), \quad \forall \boldsymbol{x}_* \in X_*(f) \tag{6.9}$$

for $n = 1, \cdots, \left\lfloor \frac{d-1}{2} \right\rfloor$, indicating that Nesterov's FGM1 and FGM2 achieve the optimal rate $O\left(\frac{1}{N^2}\right)$. However, (6.9) also illustrates the potential room for improving first-order algorithms by a constant in convergence speed.

To narrow this gap, DT [31] used a relaxation of problem (HP) to find the "optimal" choice of $\{h_{i,k}\}$ for Algorithm FO that minimizes a relaxed bound on $f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*)$ at the Nth iteration, which was found numerically to provide a twice better bound than (6.8), yet remained computationally impractical.

We next review the PEP approach for solving a relaxed version of (P) and illustrate the use of the approach by deriving new analytical bounds for the sequence $\{\boldsymbol{x}_i\}$ of FGM1 and FGM2.

6.4 A convergence bound of first-order algorithms using PEP approach

6.4.1 Review of relaxation schemes for PEP approach

This section summarizes the relaxation scheme for the PEP approach that transforms problem (P) into a tractable form [31].

Problem (P) is challenging to solve due to the (infinite-dimensional) functional constraint on f, so DT [31] cleverly relax the constraint by using the following property of convex functions f in $C_L^{1,1}$ [81, Theorem 2.1.5]:

$$\frac{1}{2L}||f'(\boldsymbol{x}) - f'(\boldsymbol{y})||^2 \leq f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle f'(\boldsymbol{y}),\ \boldsymbol{x} - \boldsymbol{y} \rangle, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{6.10}$$

DT apply this inequality (6.10) to the generated points $\{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N\}$ of Algorithm FO and any optimal point $\boldsymbol{x}_* \in X_*(f)$, and replace the functional constraint on f by a corresponding finite set of inequalities on $\{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N, \boldsymbol{x}_*\}$ using (6.10). This yields the following relaxed

version of problem (P) [31]:

$$\mathcal{B}_{\mathrm{P1}}(\boldsymbol{h}, N, L, R) \triangleq \max_{\substack{\boldsymbol{x}_0,\cdots,\boldsymbol{x}_N,\boldsymbol{x}_*\in\mathbb{R}^d,\\ \boldsymbol{g}_0,\cdots,\boldsymbol{g}_N\in\mathbb{R}^d,\\ \delta_0,\cdots,\delta_N\in\mathbb{R}}} L||\boldsymbol{x}_0-\boldsymbol{x}_*||^2\delta_N \tag{P1}$$

$$\begin{aligned}
\text{s.t. } & \boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \sum_{k=0}^{i} h_{i+1,k}||\boldsymbol{x}_0-\boldsymbol{x}_*||\boldsymbol{g}_i, \quad i=0,\cdots,N-1,\\
& \frac{1}{2}||\boldsymbol{g}_i-\boldsymbol{g}_j||^2 \le \delta_i-\delta_j-\frac{\langle \boldsymbol{g}_j,\ \boldsymbol{x}_i-\boldsymbol{x}_j\rangle}{||\boldsymbol{x}_0-\boldsymbol{x}_*||}, \quad i,j=0,\cdots,N,*,\\
& ||\boldsymbol{x}_0-\boldsymbol{x}_*|| \le R,
\end{aligned}$$

by defining

$$\begin{cases}
\delta_i \triangleq \frac{1}{L||\boldsymbol{x}_0-\boldsymbol{x}_*||^2}(f(\boldsymbol{x}_i)-f(\boldsymbol{x}_*)),\\
\boldsymbol{g}_i \triangleq \frac{1}{L||\boldsymbol{x}_0-\boldsymbol{x}_*||}f'(\boldsymbol{x}_i)
\end{cases}$$

for $i=0,\cdots,N,*$, and note that $\delta_*=0$ and $\boldsymbol{g}_*=0$. DT [31] eliminate $\{\boldsymbol{x}_0,\cdots,\boldsymbol{x}_N,\boldsymbol{x}_*\}$ using $\{\boldsymbol{g}_0,\cdots,\boldsymbol{g}_N\}$ and R in (P1), and rewrite the problem (P1) in the following form[7]:

$$\begin{aligned}
\mathcal{B}_{\mathrm{P1}}(\boldsymbol{h}, N, L, R) = \max_{\substack{\boldsymbol{G}\in\mathbb{R}^{(N+1)d},\\ \boldsymbol{\delta}\in\mathbb{R}^{N+1},\\ \boldsymbol{\nu}\in\mathbb{R}^d,||\boldsymbol{\nu}||=1}} & LR^2\delta_N\\
\text{s.t. } & \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{A}_{i,j}(\boldsymbol{h})\boldsymbol{G}\big\} \le \delta_i-\delta_j, \quad i<j=0,\cdots,N,\\
& \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{B}_{i,j}(\boldsymbol{h})\boldsymbol{G}\big\} \le \delta_i-\delta_j, \quad j<i=0,\cdots,N,\\
& \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{C}_i\boldsymbol{G}\big\} \le \delta_i, \quad i=0,\cdots,N,\\
& \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{D}_i(\boldsymbol{h})\boldsymbol{G}+\boldsymbol{\nu}\boldsymbol{u}_i^T\boldsymbol{G}\big\} \le -\delta_i, \quad i=0,\cdots,N,
\end{aligned}$$

where $i<j=0,\cdots,N$ denotes $i=0,\cdots,N-1,\ j=i+1,\cdots,N$, by defining the unit vectors[8] $\boldsymbol{u}_i=\boldsymbol{e}_{N+1,i+1}\in\mathbb{R}^{N+1}$ and $\boldsymbol{\nu}=-\frac{\boldsymbol{x}_0-\boldsymbol{x}_*}{||\boldsymbol{x}_0-\boldsymbol{x}_*||}$, the $(N+1)\times 1$ vector $\boldsymbol{\delta}=[\delta_0\cdots\delta_N]^T$,

[7] This form is a (nonhomogeneous) Quadratic Matrix Program that is introduced in [8].

[8] The vector $\boldsymbol{e}_{N,i}$ is the ith canonical basis vector in $\mathbb{R}^N$, having 1 for the ith entry and zero for all other elements.

the $(N+1)\times d$ matrix $\boldsymbol{G}=[\boldsymbol{g}_0\cdots\boldsymbol{g}_N]^T$, and the $(N+1)\times(N+1)$ symmetric matrices:

$$\begin{cases}\boldsymbol{A}_{i,j}(\boldsymbol{h})\triangleq\frac{1}{2}(\boldsymbol{u}_i-\boldsymbol{u}_j)(\boldsymbol{u}_i-\boldsymbol{u}_j)^T+\frac{1}{2}\sum_{l=i+1}^{j}\sum_{k=0}^{l-1}h_{l,k}(\boldsymbol{u}_j\boldsymbol{u}_k^T+\boldsymbol{u}_k\boldsymbol{u}_j^T),\\ \boldsymbol{B}_{i,j}(\boldsymbol{h})\triangleq\frac{1}{2}(\boldsymbol{u}_i-\boldsymbol{u}_j)(\boldsymbol{u}_i-\boldsymbol{u}_j)^T-\frac{1}{2}\sum_{l=j+1}^{i}\sum_{k=0}^{l-1}h_{l,k}(\boldsymbol{u}_j\boldsymbol{u}_k^T+\boldsymbol{u}_k\boldsymbol{u}_j^T),\\ \boldsymbol{C}_i\triangleq\frac{1}{2}\boldsymbol{u}_i\boldsymbol{u}_i^T,\\ \boldsymbol{D}_i(\boldsymbol{h})\triangleq\frac{1}{2}\boldsymbol{u}_i\boldsymbol{u}_i^T+\frac{1}{2}\sum_{j=1}^{i}\sum_{k=0}^{j-1}h_{j,k}(\boldsymbol{u}_i\boldsymbol{u}_k^T+\boldsymbol{u}_k\boldsymbol{u}_i^T).\end{cases}\tag{6.11}$$

DT [31] further relax the problem by discarding some constraints as:

$$\begin{aligned}\mathcal{B}_{\mathrm{P2}}(\boldsymbol{h},N,L,R)\triangleq\max_{\substack{\boldsymbol{G}\in\mathbb{R}^{(N+1)d},\\ \boldsymbol{\delta}\in\mathbb{R}^{N+1}}}\ & LR^2\delta_N\\ \text{s.t. }\ & \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{A}_{i-1,i}(\boldsymbol{h})\boldsymbol{G}\big\}\le\delta_{i-1}-\delta_i,\quad i=1,\cdots,N,\\ & \mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{D}_i(\boldsymbol{h})\boldsymbol{G}+\boldsymbol{\nu}\boldsymbol{u}_i^T\boldsymbol{G}\big\}\le-\delta_i,\quad i=0,\cdots,N,\end{aligned}\tag{P2}$$

for any[9] given unit vector $\boldsymbol{\nu}\in\mathbb{R}^d$.

DT [31] finally use a duality approach on (P2). Replacing $\max_{\boldsymbol{G},\boldsymbol{\delta}}LR^2\delta_N$ by $\min_{\boldsymbol{G},\boldsymbol{\delta}}-\delta_N$ for convenience, the Lagrangian of the corresponding constrained minimization problem (P2) becomes the following separable function in $(\boldsymbol{\delta},\boldsymbol{G})$:

$$\mathcal{L}(\boldsymbol{G},\boldsymbol{\delta},\boldsymbol{\lambda},\boldsymbol{\tau};\boldsymbol{h})=\mathcal{L}_1(\boldsymbol{\delta},\boldsymbol{\lambda},\boldsymbol{\tau})+\mathcal{L}_2(\boldsymbol{G},\boldsymbol{\lambda},\boldsymbol{\tau};\boldsymbol{h}),$$

where

$$\begin{cases}\mathcal{L}_1(\boldsymbol{\delta},\boldsymbol{\lambda},\boldsymbol{\tau})\triangleq-\delta_N+\sum_{i=1}^{N}\lambda_i(\delta_i-\delta_{i-1})+\sum_{i=0}^{N}\tau_i\delta_i,\\ \mathcal{L}_2(\boldsymbol{G},\boldsymbol{\lambda},\boldsymbol{\tau};\boldsymbol{h})\triangleq\sum_{i=1}^{N}\lambda_i\,\mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{A}_{i-1,i}(\boldsymbol{h})\boldsymbol{G}\big\}+\sum_{i=0}^{N}\tau_i\,\mathsf{tr}\big\{\boldsymbol{G}^T\boldsymbol{D}_i(\boldsymbol{h})\boldsymbol{G}+\boldsymbol{\nu}\boldsymbol{u}_i^T\boldsymbol{G}\big\},\end{cases}$$

with dual variables $\boldsymbol{\lambda}=(\lambda_1,\cdots,\lambda_N)^T\in\mathbb{R}_+^N$ and $\boldsymbol{\tau}=(\tau_0,\cdots,\tau_N)^T\in\mathbb{R}_+^{N+1}$. The corresponding dual function is defined as

$$H(\boldsymbol{\lambda},\boldsymbol{\tau};\boldsymbol{h})=\min_{\boldsymbol{\delta}\in\mathbb{R}^{N+1}}\mathcal{L}_1(\boldsymbol{\delta},\boldsymbol{\lambda},\boldsymbol{\tau})+\min_{\boldsymbol{G}\in\mathbb{R}^{(N+1)d}}\mathcal{L}_2(\boldsymbol{G},\boldsymbol{\lambda},\boldsymbol{\tau};\boldsymbol{h}).\tag{6.12}$$

[9] Problem (P1) is invariant under any orthogonal transformation, so DT [31] assume that $\boldsymbol{x}_0-\boldsymbol{x}_*=-\boldsymbol{\nu}||\boldsymbol{x}_0-\boldsymbol{x}_*||$ for any given unit vector $\boldsymbol{\nu}\in\mathbb{R}^d$ without loss of generality.

Here $\min_{\boldsymbol{\delta}} \mathcal{L}_1(\boldsymbol{\delta}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = 0$ for any $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$, where

$$\Lambda = \left\{ (\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \mathbb{R}_+^N \times \mathbb{R}_+^{N+1} \ : \ \begin{array}{l} \tau_0 = \lambda_1, \ \lambda_N + \tau_N = 1 \\ \lambda_i - \lambda_{i+1} + \tau_i = 0, \ i = 1, \cdots, N-1 \end{array} \right\}, \tag{6.13}$$

and $-\infty$ otherwise. Using [31, Lemma 1][10] and the lemma in [11, p. 163][11], the dual function (6.12) for any given unit vector $\boldsymbol{\nu} \in \mathbb{R}^d$ reduces to

$$\begin{aligned} H(\boldsymbol{\lambda}, \boldsymbol{\tau}; \boldsymbol{h}) &= \min_{\boldsymbol{w} \in \mathbb{R}^{N+1}} \left\{ \boldsymbol{w}^T \boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) \boldsymbol{w} + \boldsymbol{\tau}^T \boldsymbol{w} \right\} \\ &= \max_{\gamma \in \mathbb{R}} \left\{ -\frac{1}{2}\gamma \ : \ \boldsymbol{w}^T \boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) \boldsymbol{w} + \boldsymbol{\tau}^T \boldsymbol{w} \geq -\frac{1}{2}\gamma, \ \forall \boldsymbol{w} \in \mathbb{R}^{N+1} \right\} \\ &= \max_{\gamma \in \mathbb{R}} \left\{ -\frac{1}{2}\gamma \ : \ \begin{pmatrix} \boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) & \frac{1}{2}\boldsymbol{\tau} \\ \frac{1}{2}\boldsymbol{\tau}^T & \frac{1}{2}\gamma \end{pmatrix} \succeq 0 \right\} \end{aligned} \tag{6.14}$$

for any given $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$, where DT [31] define the following $(N+1) \times (N+1)$ matrix using the definition of $\boldsymbol{A}_{i-1,i}(\boldsymbol{h})$ and $\boldsymbol{D}_i(\boldsymbol{h})$ in (6.11):

$$\begin{aligned} \boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) =& \sum_{i=1}^{N} \lambda_i \boldsymbol{A}_{i-1,i}(\boldsymbol{h}) + \sum_{i=0}^{N} \tau_i \boldsymbol{D}_i(\boldsymbol{h}) \\ =& \frac{1}{2} \sum_{i=1}^{N} \lambda_i (\boldsymbol{u}_{i-1} - \boldsymbol{u}_i)(\boldsymbol{u}_{i-1} - \boldsymbol{u}_i)^T + \frac{1}{2} \sum_{i=0}^{N} \tau_i \boldsymbol{u}_i \boldsymbol{u}_i^T \\ &+ \frac{1}{2} \sum_{i=1}^{N} \sum_{k=0}^{i-1} \left(\lambda_i h_{i,k} + \tau_i \sum_{j=k+1}^{i} h_{j,k} \right) (\boldsymbol{u}_i \boldsymbol{u}_k^T + \boldsymbol{u}_k \boldsymbol{u}_i^T). \end{aligned} \tag{6.15}$$

In short, using the dual approach on the problem (P2) yields the following bound:

$$\mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, L, R) \triangleq \min_{\substack{\boldsymbol{\lambda} \in \mathbb{R}^N, \\ \boldsymbol{\tau} \in \mathbb{R}^{N+1}, \\ \gamma \in \mathbb{R}}} \left\{ \frac{1}{2} L R^2 \gamma \ : \ \begin{pmatrix} \boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) & \frac{1}{2}\boldsymbol{\tau} \\ \frac{1}{2}\boldsymbol{\tau}^T & \frac{1}{2}\gamma \end{pmatrix} \succeq 0, (\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda \right\}, \tag{D}$$

recalling that we previously replaced $\max_{\boldsymbol{G}, \boldsymbol{\lambda}} LR^2 \delta_N$ by $\min_{\boldsymbol{G}, \boldsymbol{\lambda}} -\delta_N$ for convenience. Problem (D) can be solved using any numerical SDP method [48] for given $\boldsymbol{h}$ and N, noting that

[10] Let $q(\boldsymbol{X}) = \mathsf{tr}\{\boldsymbol{X}^T \boldsymbol{S} \boldsymbol{X} + 2\boldsymbol{b}\boldsymbol{a}^T \boldsymbol{X}\}$ be a quadratic function, where $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, $\boldsymbol{a} \in \mathbb{R}^n$, $0 \neq \boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ is a symmetric matrix. Then $\inf_{\boldsymbol{X} \in \mathbb{R}^{n \times m}} q(\boldsymbol{X}) = \inf_{\boldsymbol{\xi} \in \mathbb{R}^n} q(\boldsymbol{\xi} \boldsymbol{b}^T)$.

[11] For $b \in \mathbb{R}^n$, $c \in \mathbb{R}$, and a symmetric matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, the inequality $\boldsymbol{w}^T \boldsymbol{S} \boldsymbol{w} + 2\boldsymbol{b}^T \boldsymbol{w} + c \geq 0$, $\forall \boldsymbol{w} \in \mathbb{R}^n$ holds if and only if the matrix $\begin{pmatrix} \boldsymbol{S} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{pmatrix}$ is positive semidefinite.

R is just a multiplicative scalar in (D).

Overall, DT [31] introduced a series of relaxations to the problem (P), eventually reaching the solvable problem (D) that provides a valid upper bound as

$$f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \leq \mathcal{B}_{\mathrm{P}}(\boldsymbol{h}, N, L, R) \leq \mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, L, R)$$

where $\boldsymbol{x}_N$ is generated by Algorithm FO with given $\boldsymbol{h}$ and N, and $||\boldsymbol{x}_0 - \boldsymbol{x}_*|| \leq R$. This bound is for a given $\boldsymbol{h}$ and later we optimize the bound over $\boldsymbol{h}$.

Solving problem (D) with a SDP method for any given coefficients $\boldsymbol{h}$ and N provides a numerical convergence bound for $f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*)$ [31] (with a given R). However, numerical bounds only partially explain the behavior of recursive algorithms in class FO for every N, such as GM, HBM, FGM1 and FGM2. An analytical bound (6.3) of a GM method, for example, was found using a specific PEP approach [31], but no other analytical bound was discussed in [31]. The next section exploits the PEP approach to reveal a new analytical bound of the sequence $\{f(\boldsymbol{x}_i)\}$ generated by FGM1 or FGM2 as an example, confirming the conjecture by DT that the primary sequence $\{\boldsymbol{x}_i\}$ achieves the same rate $O\left(\frac{1}{N^2}\right)$ as the sequence $\{\boldsymbol{y}_i\}$ [31, Conjecture 2].

6.4.2 An analytical bound for Nesterov's fast gradient methods

This section provides an analytical bound for the sequence $\{\boldsymbol{x}_i\}$ in FGM1 and FGM2.

For the $\bar{\boldsymbol{h}}$ factors in (6.6) or (6.7) of Nesterov's fast gradient methods, the following choice (inspired by Section 6.5.2) is a feasible point of problem (D):

$$\bar{\lambda}_i = \frac{t_{i-1}^2}{t_N^2}, \quad i = 1, \cdots, N, \tag{6.16}$$

$$\bar{\tau}_i = \frac{t_i}{t_N^2}, \quad i = 0, \cdots, N, \tag{6.17}$$

$$\bar{\gamma} = \frac{1}{t_N^2}, \tag{6.18}$$

with t_i in (6.4).

Lemma 5. *The choice* $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}, \bar{\gamma})$ *in* (6.16), (6.17) *and* (6.18) *is a feasible point of the problem* (D) *for the* $\bar{\boldsymbol{h}}$ *designs given in* (6.6) *or* (6.7) *that are used in Nesterov's FGM1 and FGM2.*

Proof. It is obvious that $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}) \in \Lambda$ using $t_i^2 = \sum_{k=0}^{i} t_k$ in (6.5). We next rewrite $\boldsymbol{S}(\bar{\boldsymbol{h}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}})$ using (6.7), (6.16) and (6.17) to show that the choice $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}, \bar{\gamma})$ satisfies the positive semidefinite condition in (D) for given $\bar{\boldsymbol{h}}$.

For any given $\boldsymbol{h}$ and $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$, the (i,k)-th entry of a symmetric matrix $\boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ in (6.15) can be written as

$$S_{i,k}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = \begin{cases} \frac{1}{2}\Big((\lambda_i + \tau_i)h_{i,k} + \tau_i \sum_{j=k+1}^{i-1} h_{j,k}\Big), & i = 2, \cdots, N,\ k = 0, \cdots, i-2, \\ \frac{1}{2}\left((\lambda_i + \tau_i)h_{i,k} - \lambda_i\right), & i = 1, \cdots, N,\ k = i-1, \\ \lambda_{i+1}, & i = 0, \cdots, N-1,\ k = i, \\ \frac{1}{2}, & i = N,\ k = i. \end{cases} \tag{6.19}$$

Inserting $\bar{\boldsymbol{h}}$ (6.7), $\bar{\boldsymbol{\lambda}}$ (6.16) and $\bar{\boldsymbol{\tau}}$ (6.17) into (6.19) and using $\bar{\lambda}_i + \bar{\tau}_i = \frac{t_i^2}{t_N^2}$ for $i = 1, \cdots, N$, we get

$$\begin{aligned} S_{i,k}(\bar{\boldsymbol{h}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}) &= \begin{cases} \frac{1}{2}\left(\frac{t_i^2}{t_N^2}\frac{1}{t_i}\left(t_k - \sum_{j=k+1}^{i-1} \bar{h}_{j,k}\right) + \frac{t_i}{t_N^2}\sum_{j=k+1}^{i-1} \bar{h}_{j,k}\right), & i = 2, \cdots, N,\ k = 0, \cdots, i-2, \\ \frac{1}{2}\left(\frac{t_i^2}{t_N^2}\left(1 + \frac{t_{i-1}-1}{t_i}\right) - \frac{t_{i-1}^2}{t_N^2}\right), & i = 1, \cdots, N,\ k = i-1, \\ \frac{t_i^2}{t_N^2}, & i = 0, \cdots, N-1,\ k = i, \\ \frac{1}{2}, & i = N,\ k = i, \end{cases} \\ &= \begin{cases} \frac{t_i t_k}{2t_N^2} & i = 1, \cdots, N,\ k = 0, \cdots, i-1, \\ \frac{t_i^2}{t_N^2}, & i = 0, \cdots, N-1,\ k = i, \\ \frac{t_N^2}{2t_N^2}, & i = N,\ k = i, \end{cases} \\ &= \frac{1}{2t_N^2}\left(\boldsymbol{t}\,\boldsymbol{t}^T + \mathsf{diag}\big\{(\check{\boldsymbol{t}}^T, 0)\big\}\right), \end{aligned}$$

where $\boldsymbol{t} = (t_0, \cdots, t_N)^T$ and $\check{\boldsymbol{t}} = (t_0^2, \cdots, t_{N-1}^2)^T$. The second equality uses $t_i^2 - t_i - t_{i-1}^2 = 0$ in (6.5), and $\mathsf{diag}\{\boldsymbol{t}\}$ denotes a matrix where diagonal elements are filled with elements of a vector $\boldsymbol{t}$ and zero for other elements.

Finally, using $\bar{\gamma}$ in (6.18), we have

$$\begin{aligned} \begin{pmatrix} \boldsymbol{S}(\bar{\boldsymbol{h}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}) & \frac{1}{2}\bar{\boldsymbol{\tau}} \\ \frac{1}{2}\bar{\boldsymbol{\tau}}^T & \frac{1}{2}\bar{\gamma} \end{pmatrix} &= \begin{pmatrix} \frac{1}{2t_N^2}\left(\boldsymbol{t}\,\boldsymbol{t}^T + \mathsf{diag}\big\{(\check{\boldsymbol{t}}^T, 0)\big\}\right) & \frac{1}{2t_N^2}\boldsymbol{t} \\ \frac{1}{2t_N^2}\boldsymbol{t}^T & \frac{1}{2t_N^2} \end{pmatrix} \\ &= \frac{1}{2t_N^2}\left\{ \begin{pmatrix} \boldsymbol{t} \\ 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{t} \\ 1 \end{pmatrix}^T + \mathsf{diag}\big\{(\check{\boldsymbol{t}}^T, 0, 0)\big\} \right\} \succeq 0. \end{aligned}$$

□

Using Lemma 5, we provide an analytical convergence bound for the sequence $\{\boldsymbol{x}_i\}$ of

FGM1 and FMG2.

Theorem 1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and $C_L^{1,1}$ and let $\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots \in \mathbb{R}^d$ be generated by FGM1 or FGM2. Then for $n \geq 1$,*

$$f(\boldsymbol{x}_n) - f(\boldsymbol{x}_*) \leq \frac{2L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{(n+2)^2}, \quad \forall \boldsymbol{x}_* \in X_*(f). \tag{6.20}$$

Proof. Using $\bar{\gamma}$ (6.18) and $t_N^2 \geq \frac{(N+2)^2}{4}$ from (6.5), we have

$$f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \leq \mathcal{B}_{\mathrm{D}}(\bar{\boldsymbol{h}}, N, L, R) \leq \frac{1}{2} LR^2 \bar{\gamma} \leq \frac{2LR^2}{(N+2)^2}, \quad \forall \boldsymbol{x}_* \in X_*(f) \tag{6.21}$$

for given $\bar{\boldsymbol{h}}$ in (6.6) or (6.7), based on Lemma 5. Since the coefficients $\bar{\boldsymbol{h}}$ in (6.6) or (6.7) are recursive and do not depend on a given N, we can extend (6.21) for all iterations ($n \geq 1$). Finally, we let $R = ||\boldsymbol{x}_0 - \boldsymbol{x}_*||$. □

Theorem 1 illustrates using the PEP approach to find an analytical bound for an algorithm in class FO. Note that we verified numerically that the choice $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}, \bar{\gamma})$ in (6.16), (6.17) and (6.18) is not an optimal solution of (D) for given $\bar{\boldsymbol{h}}$ in (6.6) or (6.7). However this feasible point $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\tau}}, \bar{\gamma})$ provides a valid upper bound for the sequence $\{\boldsymbol{x}_i\}$ of FGM1 and FGM2 as shown in Theorem 1 that is similar to (6.8) and verifies DT's conjecture [31, Conjecture 2].

The next section reviews DT's analysis [31] of the relaxed convergence bound $\mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, L, R)$ and the corresponding numerically optimized step coefficients in the class of first-order methods, using the (relaxed) PEP approach. Then, we explicitly show that the algorithm achieves a convergence bound that is twice as fast as (6.8) and (6.20).

6.5 A convergence bound for the optimized first-order algorithm

6.5.1 Review of DT's numerical bound for optimized first-order algorithms

This section summarizes the numerically optimized first-order algorithms described in [31].

Having relaxed (P) in Section 6.4.1 to (D), DT proposed to optimize $\boldsymbol{h}$ by relaxing (HP) as follows:

$$\hat{\boldsymbol{h}} \triangleq \underset{\boldsymbol{h} \in \mathbb{R}^{N(N+1)/2}}{\arg\min} \ \mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, L, R), \tag{HD}$$

where $\hat{\boldsymbol{h}}$ is independent of both L and R, since $\mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, L, R) = LR^2 \mathcal{B}_{\mathrm{D}}(\boldsymbol{h}, N, 1, 1)$. DT found $\hat{\boldsymbol{h}}$ numerically, as we review next.

Problem (HD) is a bilinear optimization problem in terms of $\boldsymbol{h}$ and the dual variables in (D), unlike the linear SDP problem (D). To simplify, DT [31] introduced a variable $\boldsymbol{r} = \{r_{i,k}\}_{0\le k<i\le N}$:

$$r_{i,k} = \lambda_i h_{i,k} + \tau_i \sum_{j=k+1}^{i} h_{j,k} \tag{6.22}$$

to convert (HD) into the following related linear SDP problem:

$$\hat{\boldsymbol{r}} \triangleq \underset{\boldsymbol{r}\in\mathbb{R}^{N(N+1)/2}}{\arg\min}\ \breve{\mathcal{B}}_{\mathrm{D}}(\boldsymbol{r}, N, L, R), \tag{RD}$$

where

$$\begin{aligned}
\breve{\mathcal{B}}_{\mathrm{D}}(\boldsymbol{r}, N, L, R) &\triangleq \min_{\substack{\boldsymbol{\lambda}\in\mathbb{R}^N,\\ \boldsymbol{\tau}\in\mathbb{R}^{N+1},\\ \gamma\in\mathbb{R}}} \left\{ \frac{1}{2}LR^2\gamma \ :\ \begin{pmatrix} \breve{\boldsymbol{S}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau}) & \frac{1}{2}\boldsymbol{\tau} \\ \frac{1}{2}\boldsymbol{\tau}^T & \frac{1}{2}\gamma \end{pmatrix} \succeq 0, (\boldsymbol{\lambda},\boldsymbol{\tau})\in\Lambda \right\}, \\
\breve{\boldsymbol{S}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau}) &\triangleq \frac{1}{2}\sum_{i=1}^{N}\lambda_i(\boldsymbol{u}_{i-1}-\boldsymbol{u}_i)(\boldsymbol{u}_{i-1}-\boldsymbol{u}_i)^T + \frac{1}{2}\sum_{i=0}^{N}\tau_i\boldsymbol{u}_i\boldsymbol{u}_i^T + \frac{1}{2}\sum_{i=1}^{N}\sum_{k=0}^{i-1} r_{i,k}(\boldsymbol{u}_i\boldsymbol{u}_k^T + \boldsymbol{u}_k\boldsymbol{u}_i^T).
\end{aligned} \tag{6.23}$$

Then, a solution $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ of linear (RD) for a given N can be computed by any numerical SDP method [48]. DT showed that the corresponding pair $(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ with the following $\hat{\boldsymbol{h}}$:

$$\hat{h}_{i,k} = \begin{cases} \frac{\hat{r}_{i,k} - \hat{\tau}_i \sum_{j=k+1}^{i-1} \hat{h}_{j,k}}{\hat{\lambda}_i + \hat{\tau}_i}, & \hat{\lambda}_i + \hat{\tau}_i \neq 0 \\ 0, & \text{otherwise,} \end{cases} \tag{6.24}$$

for $i = 1, \cdots, N,\ k = 0, \cdots, i-1$ becomes a solution of (HD) [31, Theorem 3],[12] where both (HD) and (RD) achieve the same optimal value, *i.e.*, $\mathcal{B}_{\mathrm{D}}(\hat{\boldsymbol{h}}, N, L, R) = \breve{\mathcal{B}}_{\mathrm{D}}(\hat{\boldsymbol{r}}, N, L, R)$.

The numerical results for problem (HD) in [31] provided a convergence bound that is about two-times better than that of Nesterov's fast gradient methods for a couple of choices of N in [31, Tables 1 and 2]. However, numerical calculations cannot verify the acceleration for all N, and SDP computation for solving (RD) becomes expensive for large N. In the next section, we analytically solve problem (HD), which is our first main contribution.

[12] Theorem 3 in [31] that is derived from (6.22) has typos that we fixed in (6.24).

6.5.2 An analytical bound for the optimized first-order algorithm

This section provides an analytical solution of (HD) by reformulating (RD) into a form that is tractable to solve using Karush-Kuhn-Tucker (KKT) conditions.

We first find an equivalent form of the dual function $H(\boldsymbol{\lambda}, \boldsymbol{\tau}; \boldsymbol{h})$ in (6.12) that differs from (6.14) by using the following equality:

$$S_{N,N}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = \frac{1}{2} \text{ for any } (\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda, \tag{6.25}$$

i.e., the (N, N)-th entry of $\boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ in (6.15) and (6.19) is $\frac{1}{2}$ for any $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$. Hereafter we use the notation

$$\boldsymbol{S}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) := \begin{pmatrix} \boldsymbol{Q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) & \boldsymbol{q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau}) \\ \boldsymbol{q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})^T & \frac{1}{2} \end{pmatrix}, \quad \boldsymbol{w} := \begin{pmatrix} \check{\boldsymbol{w}} \\ w_N \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\tau} := \begin{pmatrix} \check{\boldsymbol{\tau}} \\ \tau_N \end{pmatrix}, \tag{6.26}$$

where $\boldsymbol{Q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ is a $N \times N$ symmetric matrix, $\boldsymbol{q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$, $\check{\boldsymbol{w}}$ and $\check{\boldsymbol{\tau}}$ are $N \times 1$ vectors, and w_N and τ_N are scalars. We omit the arguments $(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ in $\boldsymbol{Q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ and $\boldsymbol{q}(\boldsymbol{h}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ for notational simplicity in the next derivation. For any given $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$, we rewrite $H(\boldsymbol{\lambda}, \boldsymbol{\tau}; \boldsymbol{h})$ in (6.12) and (6.14) as follows:

$$\begin{aligned}
H(\boldsymbol{\lambda}, \boldsymbol{\tau}; \boldsymbol{h}) &= \min_{\boldsymbol{w} \in \mathbb{R}^{N+1}} \left\{ \check{\boldsymbol{w}}^T \boldsymbol{Q} \check{\boldsymbol{w}} + \check{\boldsymbol{\tau}}^T \check{\boldsymbol{w}} + 2 \check{\boldsymbol{w}}^T \boldsymbol{q} w_N + \frac{1}{2} w_N^2 + \tau_N w_N \right\} \\
&= \min_{\boldsymbol{w} \in \mathbb{R}^{N+1}} \left\{ \check{\boldsymbol{w}}^T \boldsymbol{Q} \check{\boldsymbol{w}} + \check{\boldsymbol{\tau}}^T \check{\boldsymbol{w}} + \frac{1}{2} (w_N + 2 \check{\boldsymbol{w}}^T \boldsymbol{q} + \tau_N)^2 - \frac{1}{2} (2 \check{\boldsymbol{w}}^T \boldsymbol{q} + \tau_N)^2 \right\} \\
&= \min_{\check{\boldsymbol{w}} \in \mathbb{R}^N} \left\{ \check{\boldsymbol{w}}^T \boldsymbol{Q} \check{\boldsymbol{w}} + \check{\boldsymbol{\tau}}^T \check{\boldsymbol{w}} - 2 \check{\boldsymbol{w}}^T \boldsymbol{q} \boldsymbol{q}^T \check{\boldsymbol{w}} - 2 \check{\boldsymbol{w}}^T \boldsymbol{q} \tau_N - \frac{1}{2} \tau_N^2 \right\} \\
&= \min_{\check{\boldsymbol{w}} \in \mathbb{R}^N} \left\{ \check{\boldsymbol{w}}^T (\boldsymbol{Q} - 2 \boldsymbol{q} \boldsymbol{q}^T) \check{\boldsymbol{w}} + (\check{\boldsymbol{\tau}} - 2 \boldsymbol{q} \tau_N)^T \check{\boldsymbol{w}} - \frac{1}{2} \tau_N^2 \right\} \\
&= \max_{\gamma \in \mathbb{R}} \left\{ -\frac{1}{2} \gamma \ : \ \check{\boldsymbol{w}}^T (\boldsymbol{Q} - 2 \boldsymbol{q} \boldsymbol{q}^T) \check{\boldsymbol{w}} + (\check{\boldsymbol{\tau}} - 2 \boldsymbol{q} \tau_N)^T \check{\boldsymbol{w}} - \frac{1}{2} \tau_N^2 \geq -\frac{1}{2} \gamma, \ \forall \check{\boldsymbol{w}} \in \mathbb{R}^N \right\} \\
&= \max_{\gamma \in \mathbb{R}} \left\{ -\frac{1}{2} \gamma \ : \ \begin{pmatrix} \boldsymbol{Q} - 2 \boldsymbol{q} \boldsymbol{q}^T & \frac{1}{2} (\check{\boldsymbol{\tau}} - 2 \boldsymbol{q} \tau_N) \\ \frac{1}{2} (\check{\boldsymbol{\tau}} - 2 \boldsymbol{q} \tau_N)^T & \frac{1}{2} (\gamma - \tau_N^2) \end{pmatrix} \succeq 0 \right\},
\end{aligned} \tag{6.27}$$

where the third equality comes from minimizing the function with respect to w_N, and the last equality follows from a simple lemma [11, p. 163] in footnote 11.

Using (6.27) instead of (6.14) for the function $H(\boldsymbol{\lambda}, \boldsymbol{\tau}; \boldsymbol{h})$ and introducing the variable $\boldsymbol{r}$

in (6.22) leads to the following optimization problem that is equivalent to (RD):

$$\hat{\boldsymbol{r}} = \underset{\boldsymbol{r}\in\mathbb{R}^{N(N+1)/2}}{\arg\min} \ \breve{\mathcal{B}}_{\mathrm{D1}}(\boldsymbol{r}, N, L, R), \tag{RD1}$$

where

$$\breve{\mathcal{B}}_{\mathrm{D1}}(\boldsymbol{r}, N, L, R) \triangleq \min_{\substack{\boldsymbol{\lambda}\in\mathbb{R}^N,\\ \boldsymbol{\tau}\in\mathbb{R}^{N+1},\\ \gamma\in\mathbb{R}}} \left\{ \frac{1}{2}LR^2\gamma \ : \ \begin{pmatrix} \breve{\boldsymbol{Q}} - 2\breve{\boldsymbol{q}}\breve{\boldsymbol{q}}^T & \frac{1}{2}(\check{\boldsymbol{\tau}} - 2\breve{\boldsymbol{q}}\tau_N) \\ \frac{1}{2}(\check{\boldsymbol{\tau}} - 2\breve{\boldsymbol{q}}\tau_N)^T & \frac{1}{2}(\gamma - \tau_N^2) \end{pmatrix} \succeq 0, \ (\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda \right\},$$

$$\begin{aligned}
\breve{\boldsymbol{Q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau}) =& \frac{1}{2}\sum_{i=1}^{N-1} \lambda_i (\check{\boldsymbol{u}}_{i-1} - \check{\boldsymbol{u}}_i)(\check{\boldsymbol{u}}_{i-1} - \check{\boldsymbol{u}}_i)^T + \frac{1}{2}\lambda_N \check{\boldsymbol{u}}_{N-1}\check{\boldsymbol{u}}_{N-1}^T \\
&+ \frac{1}{2}\sum_{i=0}^{N-1} \tau_i \check{\boldsymbol{u}}_i \check{\boldsymbol{u}}_i^T + \frac{1}{2}\sum_{i=1}^{N-1}\sum_{k=0}^{i-1} r_{i,k}(\check{\boldsymbol{u}}_i\check{\boldsymbol{u}}_k^T + \check{\boldsymbol{u}}_k\check{\boldsymbol{u}}_i^T), \qquad (6.28)\\
\breve{\boldsymbol{q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau}) =& \frac{1}{2}\sum_{k=0}^{N-2} r_{N,k}\check{\boldsymbol{u}}_k, + \frac{1}{2}(r_{N,N-1} - \lambda_N)\check{\boldsymbol{u}}_{N-1} \qquad (6.29)
\end{aligned}$$

for $\check{\boldsymbol{u}}_i = \boldsymbol{e}_{N,i+1} \in \mathbb{R}^N$. We omit the arguments $(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ in $\breve{\boldsymbol{Q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ and $\breve{\boldsymbol{q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ for notational simplicity. Unlike (RD), we observe that the new equivalent form (RD1) has a feasible point at the boundary of the positive semidefinite condition, and we will later show that the point is indeed a solution of both (RD) and (RD1).

Lemma 6. *The choice of* $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$*:*

$$\hat{r}_{i,k} = \begin{cases} \frac{4\theta_i\theta_k}{\theta_N^2}, & i = 2, \cdots, N-1, \ k = 0, \cdots, i-2, \\ \frac{4\theta_i\theta_{i-1}}{\theta_N^2} + \frac{2\theta_{i-1}^2}{\theta_N^2}, & i = 1, \cdots, N-1, \ k = i-1, \\ \frac{2\theta_k}{\theta_N}, & i = N, \ k = 0, \cdots, i-2, \\ \frac{2\theta_{N-1}}{\theta_N} + \frac{2\theta_{N-1}^2}{\theta_N^2}, & i = N, \ k = i-1, \end{cases} \tag{6.30}$$

$$\hat{\lambda}_i = \frac{2\theta_{i-1}^2}{\theta_N^2}, \quad i = 1, \cdots, N, \tag{6.31}$$

$$\hat{\tau}_i = \begin{cases} \frac{2\theta_i}{\theta_N^2}, & i = 0, \cdots, N-1, \\ 1 - \frac{2\theta_{N-1}^2}{\theta_N^2} = \frac{1}{\theta_N}, & i = N, \end{cases} \tag{6.32}$$

$$\hat{\gamma} = \frac{1}{\theta_N^2}, \tag{6.33}$$

is a feasible point of both (RD) *and* (RD1)*, where*

$$\theta_i = \begin{cases} 1, & i = 0, \\ \frac{1+\sqrt{1+4\theta_{i-1}^2}}{2}, & i = 1, \cdots, N-1, \\ \frac{1+\sqrt{1+8\theta_{i-1}^2}}{2} & i = N. \end{cases} \tag{6.34}$$

Proof. The following set of conditions are sufficient for the feasible conditions of (RD1):

$$\begin{cases} \breve{\boldsymbol{Q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = 2\breve{\boldsymbol{q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})\breve{\boldsymbol{q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})^T, \\ \breve{\boldsymbol{\tau}} = 2\breve{\boldsymbol{q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})\tau_N, \\ \gamma = \tau_N^2, \\ (\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda. \end{cases} \tag{6.35}$$

The Appendix D shows that the point $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ in (6.30), (6.31), (6.32) and (6.33) is the unique solution of (6.35) and also satisfies the feasible conditions of (RD). □

Note that the parameter θ_i (6.34) used in Lemma 6 differs from t_i (6.4) only at the last iteration N. In other words, $\{\theta_0, \cdots, \theta_{N-1}\}$ is equivalent to $\{t_0, \cdots, t_{N-1}\}$ in (6.4) satisfying (6.5), while the last parameter θ_N satisfies

$$\theta_N^2 - \theta_N - 2\theta_{N-1}^2 = 0. \tag{6.36}$$

The next lemma shows that the feasible point derived in Lemma 6 is a solution of both (RD) and (RD1).

Lemma 7. *The choice of* $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ *in* (6.30)*,* (6.31)*,* (6.32) *and* (6.33) *is a solution of both* (RD) *and* (RD1)*.*

Proof. See Appendix E using KKT conditions[13] of (RD). □

We numerically observed that the analytical solution $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ is equivalent to the numerical solution of (RD) for couple of choices of N in [31]. The optimized step coefficients $\hat{\boldsymbol{h}}$ of interest are then derived using (6.24) [31, Theorem 3] with the analytical solution $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ of (RD).

[13] We found it easier to use the KKT conditions of linear SDP (RD) for showing the optimality of the choice $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ rather than those of bilinear SDP (RD1).

Lemma 8. *The choice of* $(\hat{\boldsymbol{h}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ *in* (6.31), (6.32), (6.33) *and*

$$\hat{h}_{i+1,k} = \begin{cases} \frac{1}{\theta_{i+1}} \left(2\theta_k - \sum_{j=k+1}^{i} \hat{h}_{j,k} \right), & k = 0, \cdots, i-1, \\ 1 + \frac{2\theta_i - 1}{\theta_{i+1}}, & k = i, \end{cases} \tag{6.37}$$

for $i = 0, \cdots, N-1$ *with* θ_i *in* (6.34) *is a solution of* (HD).

Proof. Inserting $\hat{\boldsymbol{r}}$ (6.30), $\hat{\boldsymbol{\lambda}}$ (6.31) and $\hat{\boldsymbol{\tau}}$ (6.32) into (6.24), and noting that $\hat{\lambda}_i + \hat{\tau}_i > 0$ for $i = 1, \cdots, N$, we get

$$\begin{aligned} \hat{h}_{i,k} &= \frac{\hat{r}_{i,k} - \hat{\tau}_i \sum_{j=k+1}^{i-1} \hat{h}_{j,k}}{\hat{\lambda}_i + \hat{\tau}_i}, \quad i = 1, \cdots, N, \ k = 0, \cdots, i-1, \\ &= \begin{cases} \frac{\theta_N^2}{2\theta_i^2} \left(\frac{4\theta_i \theta_k}{\theta_N^2} - \frac{2\theta_i}{\theta_N^2} \sum_{j=k+1}^{i-1} \hat{h}_{j,k} \right), & i = 1, \cdots, N-1, \ k = 0, \cdots, i-2, \\ \frac{\theta_N^2}{2\theta_i^2} \left(\frac{4\theta_i \theta_{i-1}}{\theta_N^2} + \frac{2\theta_{i-1}^2}{\theta_N^2} \right) = \frac{2\theta_i \theta_{i-1} + \theta_i^2 - \theta_i}{\theta_i^2}, & i = 1, \cdots, N-1, \ k = i-1, \\ \frac{2\theta_k}{\theta_N} - \frac{1}{\theta_N} \sum_{j=k+1}^{N-1} \hat{h}_{j,k}, & i = N, \ k = 0, \cdots, i-2, \\ \frac{2\theta_{N-1}}{\theta_N} + \frac{2\theta_{N-1}^2}{\theta_N^2} = \frac{2\theta_N \theta_{N-1} + \theta_N^2 - \theta_N}{\theta_N^2}, & i = N, \ k = i-1, \end{cases} \end{aligned}$$

which is equivalent to (6.37). From [31, Theorem 3], the corresponding $(\hat{\boldsymbol{h}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$ becomes a solution of (HD). □

The following theorem shows that Algorithm FO with the optimized $\hat{\boldsymbol{h}}$ (6.37) achieves a new fast convergence bound.

Theorem 2. *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be convex and* $C_L^{1,1}$ *and let* $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$ *be generated by Algorithm FO with* $\hat{\boldsymbol{h}}$ (6.37) *for a given* $N \geq 1$. *Then*

$$f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \leq \frac{L||\boldsymbol{x}_0 - \boldsymbol{x}_*||^2}{(N+1)(N+1+\sqrt{2})}, \quad \forall \boldsymbol{x}_* \in X_*(f). \tag{6.38}$$

Proof. Using $\hat{\gamma}$ (6.33) and $\theta_{N-1}^2 = t_{N-1}^2 \geq \frac{(N+1)^2}{4}$ from (6.5) and (6.34), we get

$$\begin{aligned} \hat{\gamma} = \frac{1}{\theta_N^2} &= \frac{4}{\left(1 + \sqrt{1 + 8\theta_{N-1}^2}\right)^2} \leq \frac{4}{\left(1 + \sqrt{1 + 2(N+1)^2}\right)^2} \\ &\leq \frac{2}{(N+1)^2 + \sqrt{2}(N+1) + 1} \leq \frac{2}{(N+1)(N+1+\sqrt{2})}. \end{aligned}$$

Then, we have

$$f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \leq \mathcal{B}_{\mathrm{D}}(\hat{\boldsymbol{h}}, N, L, R) = \frac{1}{2} L R^2 \hat{\gamma} \leq \frac{L R^2}{(N+1)(N+1+\sqrt{2})}, \quad \forall \boldsymbol{x}_* \in X_*(f),$$

based on Lemma 8. Finally, we let $R = ||\boldsymbol{x}_0 - \boldsymbol{x}_*||$. □

Theorem 2 shows that algorithm FO with the optimized $\hat{\boldsymbol{h}}$ (6.37) decreases the function f with a bound that is twice as small as that of Nesterov's fast gradient methods in (6.8) and (6.20), confirming DT's numerical results in [31, Tables 1 and 2]. The proposed algorithm requires at most $N = \left\lceil \sqrt{\frac{L}{\epsilon}} ||\boldsymbol{x}_0 - \boldsymbol{x}_*|| \right\rceil$ iterations to achieve the desired accuracy $f(\boldsymbol{x}_N) - f(\boldsymbol{x}_*) \leq \epsilon$, while Nesterov's fast gradient methods require at most $N = \left\lceil \sqrt{\frac{2L}{\epsilon}} ||\boldsymbol{x}_0 - \boldsymbol{x}_*|| \right\rceil$, a factor of about $\sqrt{2}$-times more iterations.

The next section investigates efficient implementations of the corresponding Algorithm FO with $\hat{\boldsymbol{h}}$ (6.37).

6.6 Proposed optimized first-order algorithms

6.6.1 Analytical coefficients of the optimized first-order algorithm

Even though the analytical expression for $\hat{\boldsymbol{h}}$ in (6.37) that solves (HD) does not require an expensive SDP method, using $\hat{\boldsymbol{h}}$ in Algorithm FO would still be computationally undesirable. Noticing the similarity between (6.7) of FGM2 and (6.37), we can expect that Algorithm FO with (6.37) may have equivalent efficient form as FGM2, as described in the next section. In addition, we find an equivalent form of (6.37) that is similar to (6.6) of FGM1, so that we can find a formulation that is similar to FGM1 by analogy with how Proposition 2 shows the equivalence between (6.6) and (6.7).

Proposition 3. *The optimized* $\hat{\boldsymbol{h}}$ *in* (6.37) *has the following recursive relationship*

$$\hat{h}_{i+1,k} = \begin{cases} \frac{\theta_i - 1}{\theta_{i+1}} \hat{h}_{i,k}, & k = 0, \cdots, i-2, \\ \frac{\theta_i - 1}{\theta_{i+1}} (\hat{h}_{i,i-1} - 1), & k = i-1, \\ 1 + \frac{2\theta_i - 1}{\theta_{i+1}}, & k = i, \end{cases} \tag{6.39}$$

for $i = 0, \cdots, N-1$ *with* θ_i *in* (6.34).

Proof. We follow the induction proof of Proposition 2 showing the equivalence between (6.6) and (6.7). We use the notation $\hat{h}'_{i,k}$ for the coefficient (6.37) to distinguish from (6.39).

It is obvious that $\hat{h}'_{i+1,i} = \hat{h}_{i+1,i},\ i = 0, \cdots, N-1$, and we clearly have

$$\begin{aligned}\hat{h}'_{i+1,i-1} &= \frac{1}{\theta_{i+1}}\left(2\theta_{i-1} - \hat{h}'_{i,i-1}\right) = \frac{1}{\theta_{i+1}}\left(2\theta_{i-1} - \left(1 + \frac{2\theta_{i-1}-1}{\theta_i}\right)\right) \\ &= \frac{(2\theta_{i-1}-1)(\theta_i - 1)}{\theta_i \theta_{i+1}} = \frac{\theta_i - 1}{\theta_{i+1}}\left(\hat{h}_{i,i-1} - 1\right) = \hat{h}_{i+1,i-1}.\end{aligned}$$

for $i = 0, \cdots, N-1$.

We next use induction by assuming $\hat{h}'_{i+1,k} = \hat{h}_{i+1,k}$ for $i = 0, \cdots, n-1,\ k = 0, \cdots, i$. We then have

$$\begin{aligned}\hat{h}'_{n+1,k} &= \frac{1}{\theta_{n+1}}\left(2\theta_k - \sum_{j=k+1}^{n} \hat{h}'_{j,k}\right) = \frac{1}{\theta_{n+1}}\left(2\theta_k - \sum_{j=k+1}^{n-1} \hat{h}'_{j,k} - \hat{h}'_{n,k}\right) \\ &= \frac{1}{\theta_{n+1}}\left(2\theta_k - \left(2\theta_k - \theta_n \hat{h}'_{n,k}\right) - \hat{h}'_{n,k}\right) = \frac{\theta_n - 1}{\theta_{n+1}}\hat{h}'_{n,k} = \frac{\theta_n - 1}{\theta_{n+1}}\hat{h}_{n,k} = \hat{h}_{n+1,k}\end{aligned}$$

for $k = 1, \cdots, n-2$. □

6.6.2 Efficient formulations of optimized first-order algorithms

This section revisits the derivation in Section 6.3 to transform Algorithm FO with (6.37) or (6.39) into efficient formulations like Nesterov's fast gradient methods, leading to practical algorithms.

We first propose the following optimized gradient method, called OGM1, using (6.39) in Algorithm FO. OGM1 is computationally similar to FGM1 yet the sequence $\{\boldsymbol{x}_i\}$ generated by OGM1 achieves the fast convergence bound in Theorem 2.

Algorithm OGM1

Input: $f \in C_L^{1,1}(\mathbb{R}^d)$ convex, $\boldsymbol{x}_0 \in \mathbb{R}^d,\ \boldsymbol{y}_0 = \boldsymbol{x}_0,\ \theta_0 = 1$.

For $i = 0, \cdots, N-1$

$$\boldsymbol{y}_{i+1} = \boldsymbol{x}_i - \frac{1}{L} f'(\boldsymbol{x}_i)$$

$$\theta_{i+1} = \begin{cases} \frac{1+\sqrt{1+4\theta_i^2}}{2}, & i \le N-2 \\ \frac{1+\sqrt{1+8\theta_i^2}}{2}, & i = N-1 \end{cases}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{y}_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}}(\boldsymbol{y}_{i+1} - \boldsymbol{y}_i) + \frac{\theta_i}{\theta_{i+1}}(\boldsymbol{y}_{i+1} - \boldsymbol{x}_i)$$

Apparently, the proposed OGM1 accelerates FGM1 by using just one additional momentum term $\frac{\theta_i}{\theta_{i+1}}(\boldsymbol{y}_{i+1} - \boldsymbol{x}_i)$, and thus OGM1 is computationally efficient. Also, unlike DT's approach

that requires choosing N for using SDP solver before iterating, the proposed OGM1 need not know N in advance because the coefficients $\hat{\boldsymbol{h}}$ (or θ_i) for intermediate iterations ($i = 0, \cdots, N-1$) do not depend on N.

Proposition 4. *The points $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with* (6.39) *are identical to the respective points generated by Algorithm OGM1.*

Proof. We will use induction to show that the sequence $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with (6.39) is identical to the sequence $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm OGM1. For clarity, we use the notation $\boldsymbol{x}'_0, \cdots, \boldsymbol{x}'_N$ for Algorithm FO.

It is obvious that $\boldsymbol{x}'_0 = \boldsymbol{x}_0$, and since $\theta_0 = 1$ we get

$$\boldsymbol{x}'_1 = \boldsymbol{x}'_0 - \frac{1}{L}\hat{h}_{1,0} f'(\boldsymbol{x}'_0) = \boldsymbol{x}_0 - \frac{1}{L}\left(1 + \frac{2\theta_0 - 1}{\theta_1}\right) f'(\boldsymbol{x}_0) = \boldsymbol{y}_1 + \frac{\theta_0}{\theta_1}(\boldsymbol{y}_1 - \boldsymbol{x}_0) = \boldsymbol{x}_1.$$

Assuming $\boldsymbol{x}'_i = \boldsymbol{x}_i$ for $i = 0, \cdots, n$, we then have

$$\begin{aligned}
\boldsymbol{x}'_{n+1} =& \boldsymbol{x}'_n - \frac{1}{L}\hat{h}_{n+1,n} f'(\boldsymbol{x}'_n) - \frac{1}{L}\hat{h}_{n+1,n-1} f'(\boldsymbol{x}'_{n-1}) - \frac{1}{L}\sum_{k=0}^{n-2} \hat{h}_{n+1,k} f'(\boldsymbol{x}'_k) \\
=& \boldsymbol{x}_n - \frac{1}{L}\left(1 + \frac{2\theta_n - 1}{\theta_{n+1}}\right) f'(\boldsymbol{x}_n) - \frac{\theta_n - 1}{\theta_{n+1}}(\hat{h}_{n,n-1} - 1) f'(\boldsymbol{x}_{n-1}) - \frac{1}{L}\sum_{k=0}^{n-2} \frac{\theta_n - 1}{\theta_{n+1}} \hat{h}_{n,k} f'(\boldsymbol{x}_k) \\
=& \boldsymbol{x}_n - \frac{1}{L}\left(1 + \frac{\theta_n}{\theta_{n+1}}\right) f'(\boldsymbol{x}_n) + \frac{\theta_n - 1}{\theta_{n+1}}\left(-\frac{1}{L} f'(\boldsymbol{x}_n) + \frac{1}{L} f'(\boldsymbol{x}_{n-1}) - \frac{1}{L}\sum_{k=0}^{n-1} \hat{h}_{n,k} f'(\boldsymbol{x}_k)\right) \\
=& \boldsymbol{y}_{n+1} + \frac{\theta_n}{\theta_{n+1}}(\boldsymbol{y}_{n+1} - \boldsymbol{x}_n) + \frac{\theta_n - 1}{\theta_{n+1}}\left(-\frac{1}{L} f'(\boldsymbol{x}_n) + \frac{1}{L} f'(\boldsymbol{x}_{n-1}) + \boldsymbol{x}_n - \boldsymbol{x}_{n-1}\right) \\
=& \boldsymbol{y}_{n+1} + \frac{\theta_n - 1}{\theta_{n+1}}(\boldsymbol{y}_{n+1} - \boldsymbol{y}_n) + \frac{\theta_n}{\theta_{n+1}}(\boldsymbol{y}_{n+1} - \boldsymbol{x}_n) = \boldsymbol{x}_{n+1}
\end{aligned}$$

□

Next, we propose another efficient formulation of Algorithm FO with (6.37) that is similar to the formulation of FGM2.

Algorithm OGM2

Input: $f \in C_L^{1,1}(\mathbb{R}^d)$ convex, $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\theta_0 = 1$.

For $i = 0, \cdots, N-1$

$$\boldsymbol{y}_{i+1} = \boldsymbol{x}_i - \frac{1}{L} f'(\boldsymbol{x}_i)$$

$$\boldsymbol{z}_{i+1} = \boldsymbol{x}_0 - \frac{1}{L} \sum_{k=0}^{i} 2\theta_k f'(\boldsymbol{x}_k)$$

$$\theta_{i+1} = \begin{cases} \frac{1+\sqrt{1+4\theta_i^2}}{2}, & i \leq N-2 \\ \frac{1+\sqrt{1+8\theta_i^2}}{2}, & i = N-1 \end{cases}$$

$$\boldsymbol{x}_{i+1} = \left(1 - \frac{1}{\theta_{i+1}}\right) \boldsymbol{y}_{i+1} + \frac{1}{\theta_{i+1}} \boldsymbol{z}_{i+1}$$

The sequence $\{\boldsymbol{x}_i\}$ generated by OGM2 achieves the fast convergence bound in Theorem 2. Algorithm OGM2 doubles the weight on all previous gradients for $\{\boldsymbol{z}_i\}$ compared to FGM2, providing some intuition for its two-fold acceleration. OGM2 requires the same computation as FGM2.

Proposition 5. *The points $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with* (6.37) *are identical to the respective points generated by Algorithm OGM2.*

Proof. We will use induction to show that the sequence $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm FO with (6.37) is identical to the sequence $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_N$ generated by Algorithm OGM2. For clarity, we use the notation $\boldsymbol{x}'_0, \cdots, \boldsymbol{x}'_N$ for Algorithm FO.

It is obvious that $\boldsymbol{x}'_0 = \boldsymbol{x}_0$, and since $\theta_0 = 1$ we get

$$\boldsymbol{x}'_1 = \boldsymbol{x}'_0 - \frac{1}{L} \hat{h}_{1,0} f'(\boldsymbol{x}'_0) = \boldsymbol{x}_0 - \frac{1}{L} \left(1 + \frac{2\theta_0 - 1}{\theta_1}\right) f'(\boldsymbol{x}_0) = \boldsymbol{y}_1 + \frac{\theta_0}{\theta_1} (\boldsymbol{y}_1 - \boldsymbol{x}_0) = \boldsymbol{x}_1.$$

Assuming $\boldsymbol{x}'_i = \boldsymbol{x}_i$ for $i = 0, \cdots, n$, we then have

$$\begin{aligned}
\boldsymbol{x}'_{n+1} =& \boldsymbol{x}'_n - \frac{1}{L} \hat{h}_{n+1,n} f'(\boldsymbol{x}'_n) - \frac{1}{L} \sum_{k=0}^{n-1} \hat{h}_{n+1,k} f'(\boldsymbol{x}'_k) \\
=& \boldsymbol{x}_n - \frac{1}{L} \left(1 + \frac{2\theta_n - 1}{\theta_{n+1}}\right) f'(\boldsymbol{x}_n) - \frac{1}{L} \sum_{k=0}^{n-1} \frac{1}{\theta_{n+1}} \left(2\theta_k - \sum_{j=k+1}^{n} \hat{h}_{j,k}\right) f'(\boldsymbol{x}_k) \\
=& \left(1 - \frac{1}{\theta_{n+1}}\right) \left(\boldsymbol{x}_n - \frac{1}{L} f'(\boldsymbol{x}_n)\right) + \frac{1}{\theta_{n+1}} \left(\boldsymbol{x}_n + \frac{1}{L} \sum_{k=0}^{n-1} \sum_{j=k+1}^{n} \hat{h}_{j,k} f'(\boldsymbol{x}_k) - \frac{1}{L} \sum_{k=0}^{n} 2\theta_k f'(\boldsymbol{x}_k)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \left(1 - \frac{1}{\theta_{n+1}}\right) \boldsymbol{y}_{n+1} + \frac{1}{\theta_{n+1}} \left(\boldsymbol{x}_n + \frac{1}{L} \sum_{j=1}^{n} \sum_{k=0}^{j-1} \hat{h}_{j,k} f'(\boldsymbol{x}_k) - \frac{1}{L} \sum_{k=0}^{n} 2\theta_k f'(\boldsymbol{x}_k) \right) \\
&= \left(1 - \frac{1}{\theta_{n+1}}\right) \boldsymbol{y}_{n+1} + \frac{1}{\theta_{n+1}} \left(\boldsymbol{x}_0 - \frac{1}{L} \sum_{k=0}^{n} 2\theta_k f'(\boldsymbol{x}_k) \right) \\
&= \left(1 - \frac{1}{\theta_{n+1}}\right) \boldsymbol{y}_{n+1} + \frac{1}{\theta_{n+1}} \boldsymbol{z}_{n+1} = \boldsymbol{x}_{n+1}
\end{aligned}
$$

The fifth equality uses the telescoping sum $\boldsymbol{x}_n = \boldsymbol{x}_0 + \sum_{j=1}^{n} (\boldsymbol{x}_j - \boldsymbol{x}_{j-1})$ and (6.1) in Algorithm FO. □

Our proposed OGM1 and OGM2 methods use the parameter θ_i in (6.34) that differs from t_i in (6.4) only at the last iteration. Therfore, we conclude this section by a conjecture about convergence bounds for modified versions of OGM1 and OGM2 that simply use t_i for all iterations.

Conjecture 1. *Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots$ be the sequence generated by either OGM1 or OGM2 with t_i in (6.4), instead of θ_i in (6.34), then the sequence $\{f(\boldsymbol{x}_i)\}$ converges to $f(\boldsymbol{x}_*)$ with similar convergence bound as that of the original OGM1 or OGM2 in Theorem 2.*

6.7 Conclusion and Discussion

We proposed new optimized first-order algorithms that are twice as fast as Nesterov's methods for smooth unconstrained convex minimization, inspired by DT's recent work [31]. The proposed first-order methods are comparably efficient for implementation as Nesterov's methods. Thus it is natural to use the proposed OGM1 and OGM2 to replace Nesterov's methods in smooth unconstrained convex minimization.

The new optimized first-order algorithms lack convergence bounds for the intermediate iterations, but we conjecture that such bounds are similar as for the last iteration (N). Deriving convergence bounds for the intermediate iterations may help further understand the behavior of the proposed algorithms. In addition, just as Nesterov's fast gradient methods have been extended for nonsmooth convex minimization [10, 84], extending the proposed optimized first-order algorithms for minimizing nonsmooth convex function would be a natural direction to pursue. Last, DT's PEP approach involves a series of relaxations to make the problem solvable, so there is likely still room for improvement in optimizing first-order methods, which we leave as future work.

In addition to introducing new OGM methods in class FO, we next show the convergence behavior of the proposed OGM methods with OS methods on simulated and real CT scans,

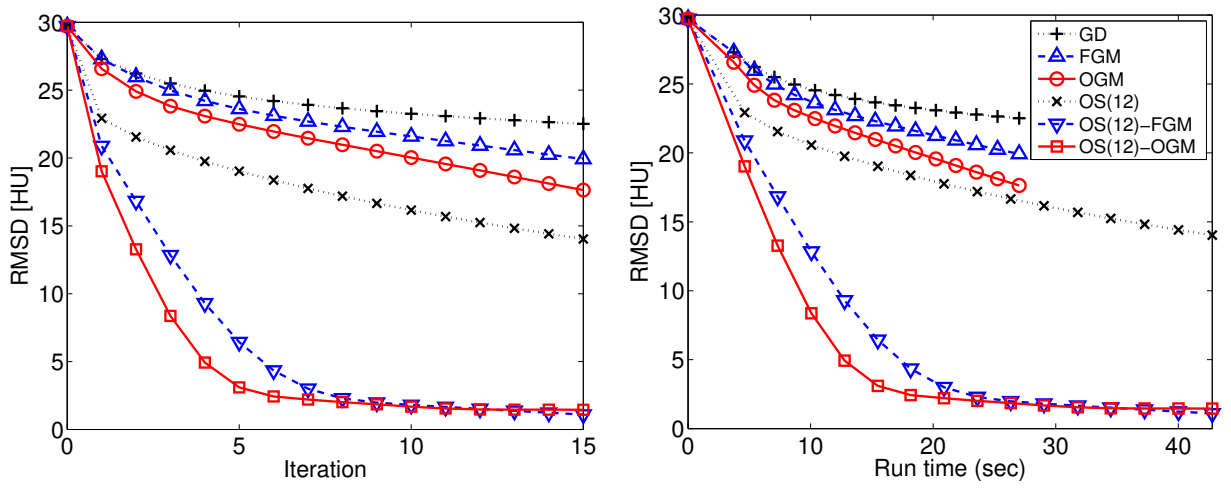
and show that they provide faster convergence speed than the OS methods when combined with Nesterov's FGM methods proposed in Chapter V.

6.8 Results

We combine the optimized gradient method (OGM) with OS methods, which we expect to converge faster than OS methods with Nesterov's momentum FGM method. We also replaced the $1/L$ factor in the OGM method with a diagonal matrix D^{-1} based on separable quadratic surrogates [2]; this D is easier to compute than the (smallest) Lipschitz constant L. We used 2D simulation data and 3D real patient data to measure the acceleration of the proposed algorithms.

6.8.1 Simulation data

We simulated 2D fan-beam CT 492×444 noisy sinogram data from a 512×512 XCAT phantom image [98]. We reconstructed a 256×256 image from the sinogram using OS methods (1 and 12 subsets) with and without momentum techniques for 15 iterations.



(a) vs. Iteration (b) vs. Run time

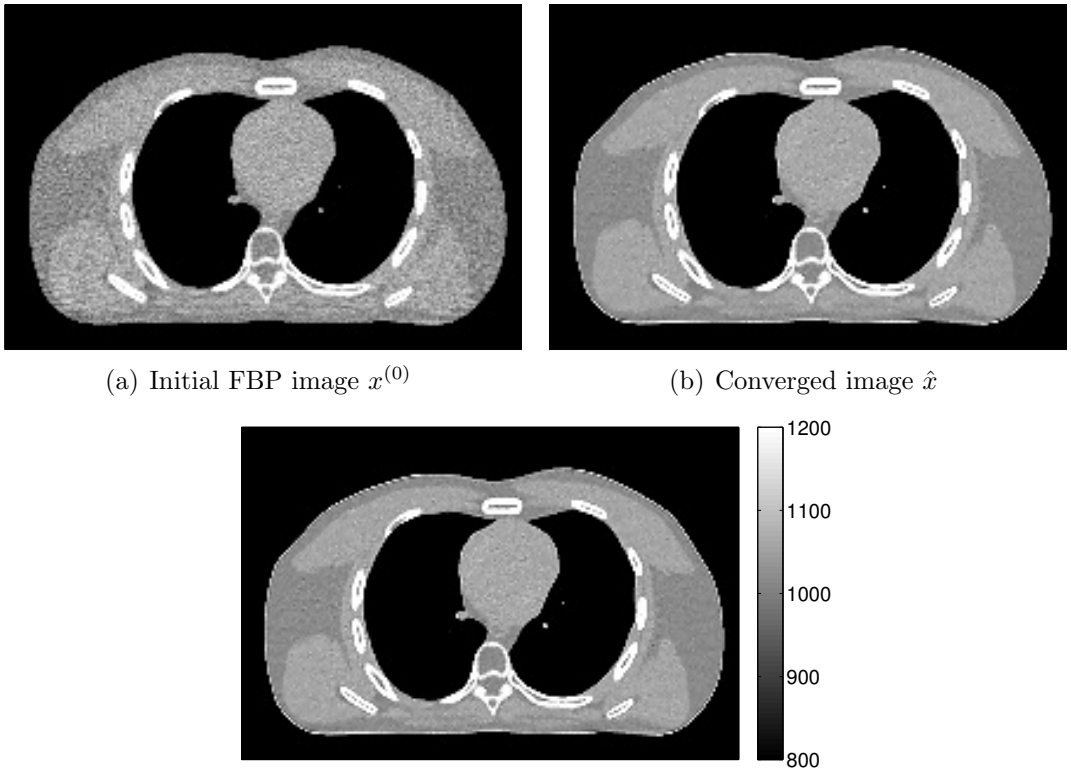
Figure 6.1: Plots of RMSD [HU] versus (a) iteration and (b) run time (sec) for OS methods using 1 and 12 subsets with and without momentum techniques. Each iteration of OS methods with 12 subsets performs 12 sub-iterations. (GD is an abbreviation for gradient descent methods, also known as gradient methods (GM).)

Fig. 6.1 illustrates the root mean square difference (RMSD) between $x^{(n)}$ and the con-

verged image $\hat{x}$ in Hounsfield Units (HU):

$$\text{RMSD}^{(n)} = \frac{||x^{(n)} - \hat{x}||}{\sqrt{N_p}} \ \text{[HU]} \tag{6.40}$$

versus both iteration and run time, to evaluate the convergence rate. The results show that two momentum techniques provide acceleration. Particularly, the proposed OGM approaches the converged image faster than Nesterov's FGM method in both iteration and run time, as expected. Even though the OGM algorithm is thus far proven to achieve fast convergence only at the final Nth iteration, the algorithm shows acceleration within all N iterations in this experiment. In Fig. 6.1, using 12 subsets in OS methods accelerated all algorithms, even though it slightly increased the computation time per iteration for executing 12 sub-iterations per each iteration.



(a) Initial FBP image $x^{(0)}$

(b) Converged image $\hat{x}$

(c) Reconstructed image $x^{(5)}$

Figure 6.2: 2D XCAT simulation: (a) an initial FBP image $x^{(0)}$, (b) a converged image $\hat{x}$, and (c) a reconstructed image $x^{(5)}$ from 5 iterations of the proposed OGM algorithm using 12 subsets.

Fig. 6.2 shows an initial filtered back-projection (FBP) image $x^{(0)}$, a converged image $\hat{x}$, and a reconstructed image from 5 iterations of the proposed OGM algorithm with OS(12)

method. The result indicates that we can reach nearby the converged image within very few iterations using the proposed algorithm.

6.8.2 Shoulder region scan data

We reconstructed $512 \times 512 \times 109$ image from a shoulder region scan $888 \times 32 \times 7146$ sinogram with pitch 0.5 described in Section 4.2.3.2.

Fig. 6.3(a) illustrates that FGM and OGM are much faster than gradient descent (GD) as expected. We can further experimentally verify that the number of iterations required for reaching the desired accurracy is about $\frac{1}{\sqrt{2}}$ less for the proposed OGM compared to Nesterov's FGM, as discussed in Section 6.5.2. Fig. 6.3(b) further shows results of GD, FGM and OGM combined with OS methods for 12 subsets, illustrating that OS combined with OGM is the fastest among all choices.

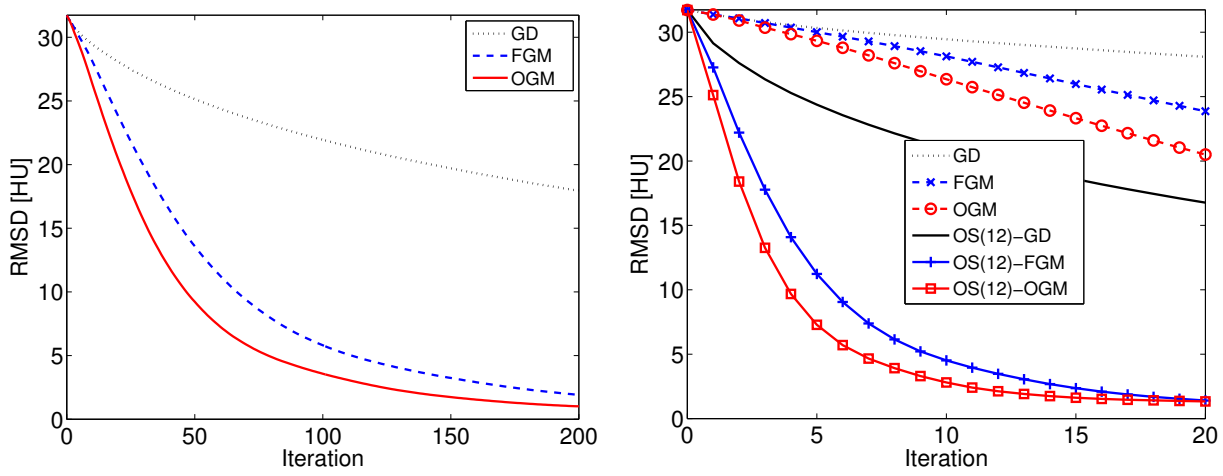


Figure 6.3: Plots of RMSD [HU] versus iteration for OS methods using 1 and 12 subsets with and without momentum techniques such as FGM and OGM. Each iteration of OS methods with 12 subsets performs 12 sub-iterations.

CHAPTER VII

Axial block coordinate descent in 3D cone-beam X-ray CT

This section develops an iterative algorithm in a different viewpoint compared to previous chapters. Here, we design an optimization algorithm that is specifically designed for 3D cone-beam X-ray CT geometry. We briefly review the 3D cone-beam geometry and the corresponding system matrix, particularly separable footprint (SF) projector [70]. Then, we propose an axial block coordinate descent (ABCD) [42] that is one instance of a general block coordinate descent (BCD) methods [49, 54]. Carefully chosen axial block based on the geometry and the projector leads to both fast convergence rate and efficient implementation.

The basic idea of this chapter has been published as a conference paper [42].

7.1 3D cone-beam X-ray CT geometry and system matrix

We provide a background of 3D cone-beam X-ray CT and discuss the advantages of using an axial block for block coordinate descent (BCD) algorithm in 3D X-ray CT.

7.1.1 3D cone-beam X-ray CT geometry

We generalize Section 2.1 to a 3D cone-beam X-ray CT geometry. Each ray in 3D cone-beam X-ray CT geometry can be characterized by (β, s, t), where (s, t) indicates the 2D coordinate of a detector, and β denotes the angle between the source and y axis. Both source-to-isocenter distance D_{s0} and isocenter-to-detector distance D_{0d} that characterize the geometry remain fixed in standard CT (see Fig. 7.1).

An axial cone-beam flat-detector CT geometry is illustrated in Fig. 7.1, where the source and the detector rotates with respect to the z-axis to acquire data. Helical geometry is used to scan many slices along z-axis than axial geometry, where the object moves in z direction while the source and detector rotates. The corresponding source trajectory of

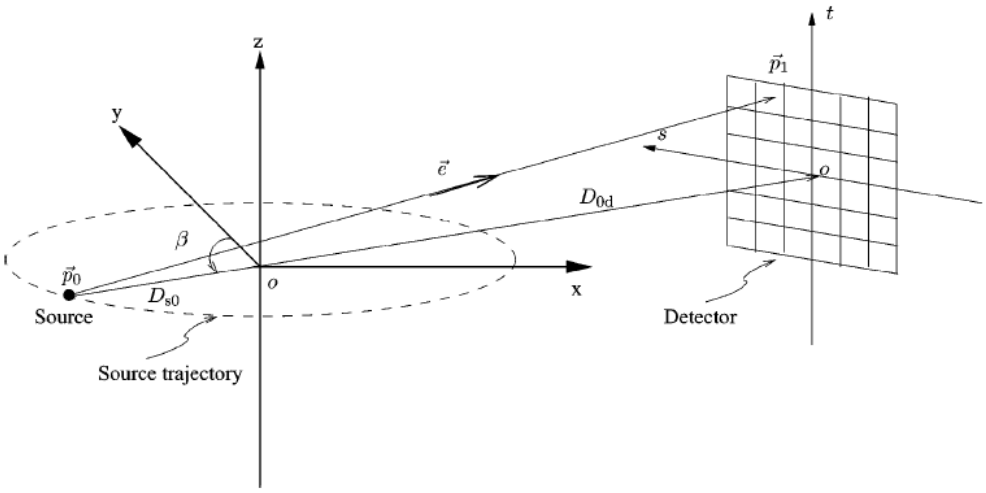


Figure 7.1: Axial cone-beam flat-detector CT geometry [70]

.

helical geometry follows a helical line as described in Fig. 3.1, in contrast to a circle trajectory in axial geometry (see Fig. 7.1).

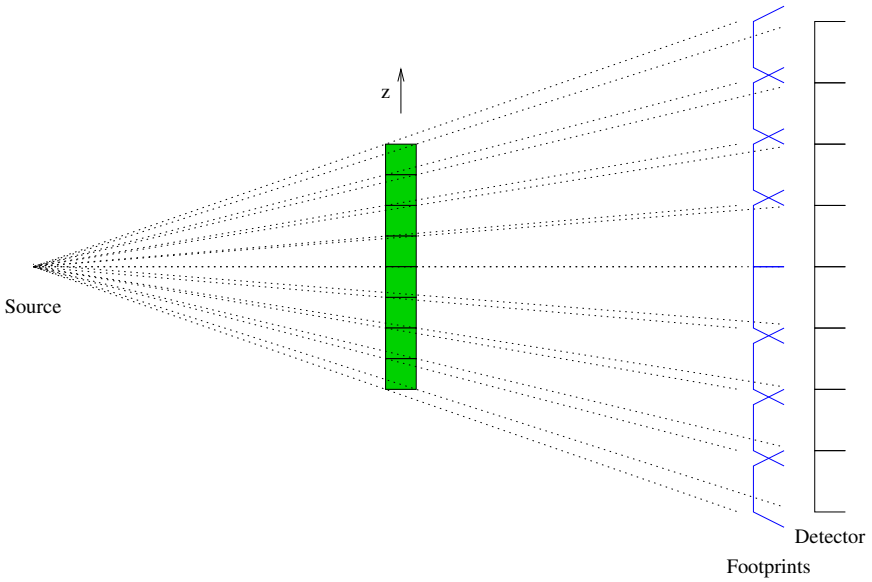


Figure 7.2: Axial footprint overlap.

A voxel has 2D footprint for each projection view (β) in 3D cone-beam geometry. Typically, the axial (t) footprints of at most three voxels which are adjacent along z-axis overlap on any given detector cell for standard 3D CT geometry, as shown in Fig. 7.2. This means that the coupling among voxels within an axial (z) column are relatively small than that

within a (highly-coupled) transaxial ($x-y$) plane. This point is importantly considered in ABCD algorithm to choose an axial block for a block coordinate descent.

We further discuss the implementation of system matrix in 3D X-ray CT. We particularly focus on SF projector [70] that works well with the proposed ABCD algorithm.

7.1.2 System matrix in 3D X-ray CT geometry

A system matrix in (2.4) can be rewritten with geometry-based parameters as below:

$$g(\beta, s, t) = \sum_{r_x, r_y, r_z} a(\beta, s, t, ; r_x, r_y, r_z) x(r_x, r_y, r_z), \tag{7.1}$$

where $a(\beta, s, t; r_x, r_y, r_z)$ represents the footprint of the voxel located at (r_x, r_y, r_z). Accurately computing the footprint is impractical, and both a separable footprint (SF) projector [70] and a distance-driven (DD) projector [21] approximate the footprint to be separable for computational efficiency. Here, we focus on SF projector.

By simplifying the derivation of SF projector [70], we can approximate the footprint as below (magnification factor has been omitted):

$$a(\beta, s, t; r_x, r_y, r_z) \approx \tilde{a}(\beta, s, t; r_x, r_y, r_z) \triangleq f_1(\beta, s; r_x, r_y) f_2(\beta, t; r_x, r_y, r_z), \tag{7.2}$$

where each $f_1(\beta, s; r_x, r_y)$ and $f_2(\beta, t; r_x, r_y, r_z)$ denote axial (t) and transaxial (s) footprint. The separability leads to very efficient implementation. For a given projection view β, a forward projection can be efficiently executed as:

$$g(\beta, s, t) \approx \tilde{g}(\beta, s, t) = \sum_{r_x, r_y} f_1(\beta, s; r_x, r_y) \sum_{r_z} f_2(\beta, t; r_x, r_y, r_z) x(r_x, r_y, r_z). \tag{7.3}$$

Similarly, back projection can be computed efficiently as:

$$x(r_x, r_y, r_z) = \sum_{s,t,\beta} a(s, t, \beta; r_x, r_y, r_z) g(s, t, \beta) \tag{7.4}$$

$$\approx \sum_{\beta} \sum_{t} f_2(\beta, t; r_x, r_y, r_z) \sum_{s} f_1(\beta, s; x, y) g(\beta, s, t). \tag{7.5}$$

These derivations indicate that a system matrix based on the separability of footprint can be efficiently implemented when the computation are executed along the axial z-axis and s-axis. So, an algorithm that updates voxels within an axial (z) column simultaneously can be implemented efficiently for a system matrix like SF projector.

Based on the discussions of the geometry and system matrix in 3D X-ray CT, we pro-

pose an axial block coordinate descent (ABCD) algorithm updating voxels within an axial column simultaneously, which is expected to achieve both fast convergence and efficient implementation.

7.2 Axial block coordinate descent (ABCD)

Considering both the modern computing architectures and the convergence rate of ICD and PCG/OS (as discussed in Section 2.3.1), a compromise between updating only one voxel at a time and updating all voxels simultaneously seems to be preferable, such as a block coordinate descent (BCD) algorithms. These algorithms have been used in statistical estimation in [49, 54], and have also been applied to statistical image reconstruction [12, 40, 41, 97]. However, the previous works of BCD for 2D tomographic image reconstruction did not have dramatic acceleration due to the high-coupling within a transaxial $(x-y)$ plane.

A group of voxels based on checker-board patterns in Fig. 7.3(a) with optimization transfer method was used in a group coordinate descent (GCD) [41]. A checker-board pattern was chosen for each voxels to be far from each other, which reduced the coupling-related curvature for optimization transfer. But the coupling remained high due to high-coupling within transaxial $(x-y)$ plane, and thus the algorithm did not have a dramatic acceleration. In other hand, a block-iterative coordinate descent (B-ICD) [12] selected a rectangular block of voxels in Fig. 7.3(b) without using separable surrogate approach, thereby avoiding the high curvatures. But inverting a dense Hessian matrix became a bottleneck. In short, a group or block of voxels within a transaxial $(x-y)$ slice in previous works for 2D CT were highly coupled, thus either high curvature for separable surrogate or computationally expensive computation for dense Hessian matrix prevented the dramatic acceleration.

We can extend these ideas straightfowardly to 3D CT geometry of our interest, but those are not promising for the same reason. Instead, we suggest to choose axial column blocks in Fig. 7.3(c) for 3D axial and helical axial cone-beam X-ray CT image reconstruction. This choice have two advantages based on Section 7.1. First, the Hessian of an axial column block is banded (as discussed in Section 7.1.1), typically penta-diagonal for standard 3D cone-beam geometry. This leads to inexpensive computation for inverting the banded Hessian [47, Chapter 5], or provides a small coupling-related curvature for optimization transfer leading to fast convergence rate. Second, the proposed ABCD algorithm can be implemented efficiently by designing it hand-in-hand with the separable footprint (SF) projector [70] (as discussed in Section 7.1.2). Preliminary simulation results show that the ABCD algorithms are comparable to ICD considering the convergence rate. Both ICD and ABCD algorithms converge faster than the optimization transfer method-based SQS algorithm.

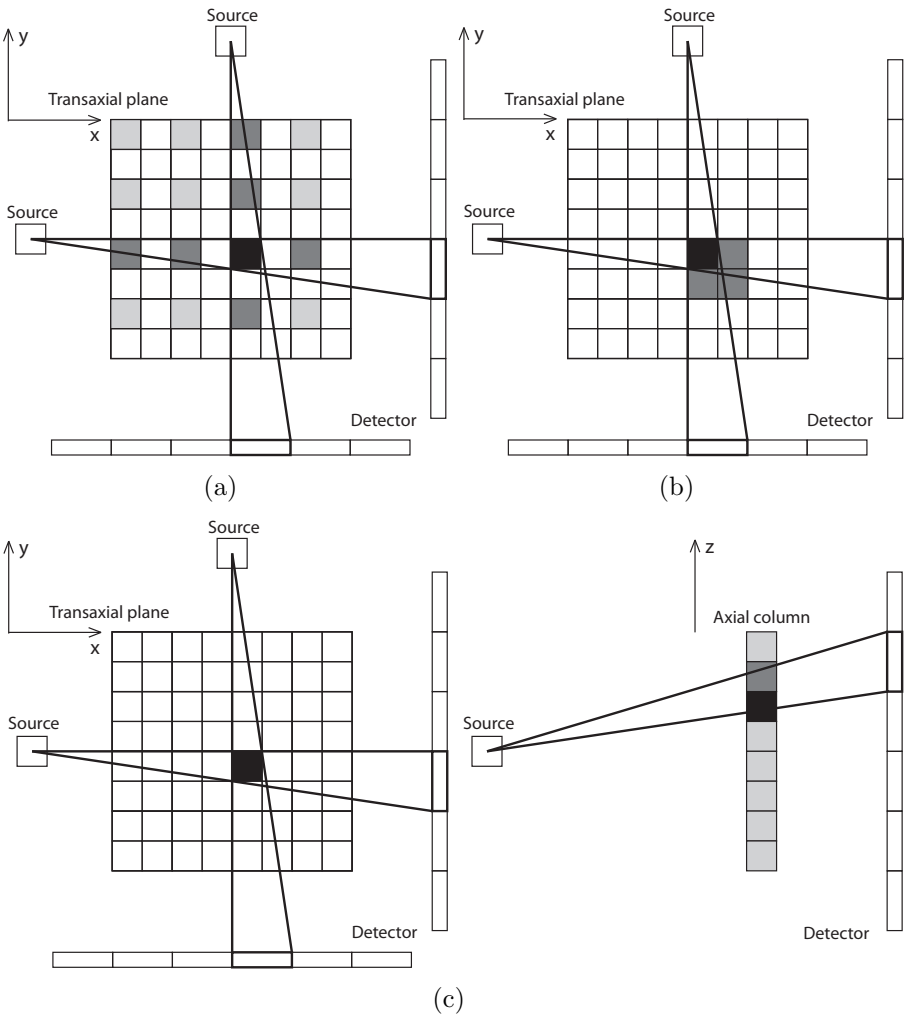


Figure 7.3: Coupling between (colored) voxels within a group in (a) GCD [41] (b) B-ICD [12] and (c) ABCD algorithms. Middle gray denotes the voxels that are coupled with a reference (black) voxel. The ratio of mid-gray voxels within a group illustrates the amount coupling within a group. GCD and B-ICD have dense Hessian matrix, while Hessian matrix in ABCD is banded. (Hessian matrix in GCD is small-eigenvalued compared with that in B-ICD.)

7.2.1 ABCD algorithm

We first review a general block coordinate descent (BCD) method [49, 54]. The voxels of image $x = (x_{b_1}, \ldots, x_{b_K})$ are (disjointly) partitioned into K sets. The BCD algorithm updates each block x_{b_k} sequentially using the most recent estimates of other blocks $\left(x_{b_1}^{(n+1)}, \ldots, x_{b_{k-1}}^{(n+1)}, x_{b_k}, x_{b_{k+1}}^{(n)}, \ldots, x_{b_K}^{(n)}\right)$ (where n counts the number of iterations), by mini-

mizing the cost function $\Psi(x)$ with respect to x_{b_k}:

$$x_{b_k}^{(n+1)} = \arg\min_{x_{b_k}} \Psi_{b_k}^{(n)}(x_{b_k})$$
$$\Psi_{b_k}^{(n)}(x_{b_k}) \triangleq \Psi\left(x_{b_1}^{(n+1)}, \ldots, x_{b_{k-1}}^{(n+1)}, x_{b_k}, x_{b_{k+1}}^{(n)}, \ldots, x_{b_K}^{(n)}\right), \tag{7.6}$$

which reduces to ICD when each block consists of a single voxel.

Since the function $\Psi_{b_k}^{(n)}(x_{b_k})$ cannot be minimized in one step, we use an optimization transfer method in Section 2.3.2 with a surrogate function $\phi_{b_k}^{(n)}(x_{b_k})$ for $\Psi_{b_k}^{(n)}(x_{b_k})$, which leads to following update step:

$$x_{b_k}^{(n+1)} = \arg\min_{x_{b_k}} \phi_{b_k}^{(n)}(x_{b_k}). \tag{7.7}$$

The convergence rate of an optimization transfer method depends on the coupling-related curvature of the surrogate function. (We define the coupling between two voxels as the inner product of two voxels' footprint on the same detector element, and also mathematically as $\sum_{i=1}^{N_d} a_{ij} a_{ij'}$ for $j \neq j'$.) Reducing the coupling-related curvature is likely to speed up the convergence rate, and our choice axial block column provides the reduced coupling as discussed in Section 7.1.

We first use a quadratic surrogate approach that leads to a non-diagonal but banded Hessian matrix $H_{b_k}^{(n)}$, usually a penta-diagonal for a standard multi-slice CT geometry. We call this algorithm as ABCD-BAND that generates a sequence $\left\{x^{(n)}\right\}$ by

$$x_{b_k}^{(n+1)} = x_{b_k}^{(n)} - \left[H_{b_k}^{(n)}\right]^{-1} \nabla \Psi_{b_k}^{(n)}\left(x_{b_k}^{(n)}\right), \tag{7.8}$$

where the matrix $H_{b_k}^{(n)}$ can be inverted in $O(n)$ operations [47, Chapter 5] that is much faster than $O(n^3)$ for a dense Hessian in B-ICD algorithm [12].

We also investigated two other methods for updating the kth block in one step, use the optimization transfer methods based on QSQS in Section 4.3.2 and SQS in Section 4.1. Since a banded Hessian $H_{b_k}^{(n)}$ has most of its energy in the three main diagonal elements, the QSQS approach that provides a tridiagonal Hessian matrix is a reasonable choice of surrogates that may reduce the computation. This is because inverting a tridiagonal Hessian $T_{b_k}^{(n)}$ require $O(n)$ operations with relatively small constant compared to inverting $H_{b_k}^{(n)}$. We refer this method as ABCD-QSQS that leads to:

$$x_{b_k}^{(n+1)} = x_{b_k}^{(n)} - \left[T_{b_k}^{(n)}\right]^{-1} \nabla \Psi_{b_k}^{(n)}\left(x_{b_k}^{(n)}\right). \tag{7.9}$$

Similarly, we used SQS method in (7.7), which may slow down the convergence rate slightly compared to two previous choices (7.8) and (7.9), since the coupling of an axial group of pixels is relatively small, and the corresponding diagonal Hessian $D_{b_k}^{(n)}$ simplifies the update. This algorithm is named as ABCD-SQS which is similar to (4.57):

$$x_{b_k}^{(n+1)} = x_{b_k}^{(n)} - \left[D_{b_k}^{(n)}\right]^{-1} \nabla \Psi_{b_k}^{(n)} \left(x_{b_k}^{(n)}\right). \tag{7.10}$$

Note that the Hessian matrix $D_{b_k}^{(n)}$ is small-eigenvalued, while the surrogates in GCD [41] have high curvatures.

So far we only considered the data fit term $L(x)$. The typical edge-preserving regularizers used for 3D CT [103] use the immediate neighboring voxels. In such regularizers, voxels in an axial column are coupled to those immediately above and immediately below, and thus the Hessian matrix for the regularizer within an axial column block is always tridiagonal. This is another benefit of using an axial column block, which only requires quadratic surrogate approach to maintain the nature of ABCD-BAND and ABCD-QSQS, while ABCD-SQS requires additional separable surrogate for the regularizer.

The difference among three ABCD algorithms (7.8), (7.9), and (7.10) is the use of different surrogate approaches and corresponding Hessian matrix that characterizes the convergence behavior and computational efficiency of three ABCD methods. Three different Hessian matrices satisfy

$$H_{b_k}^{(n)} \prec T_{b_k}^{(n)} \prec D_{b_k}^{(n)}, \tag{7.11}$$

which means that the banded Hessian matrix is small eigen-valued that leads to fast convergence rate, while inverting the banded matrix is relatively expensive than other two choices. So, we should study the trade-off among three choices.

7.2.2 Prelimary simulation results

We implemented the three ABCD algorithms in Matlab for PWLS cost function, and also ICD ($K = N_p$) and SQS ($K = 1$) methods for comparison. The simulated 3D image was limited to a small size, $64 \times 64 \times 16$, because Matlab is a slow interpreted language. This simulation mainly focused on the convergence rate per iteration. We left the comparison of run time per iteration as future work, but we expect the run time per iteration will obey the following inequalities:

$$\text{ICD} > \text{ABCD-BAND} > \text{ABCD-QSQS} > \text{ABCD-SQS} > \text{SQS}. \tag{7.12}$$

On the other hand, the convergence rate will also approximately follow (7.12), although ABCD-BAND might converge in fewer iterations than ICD.

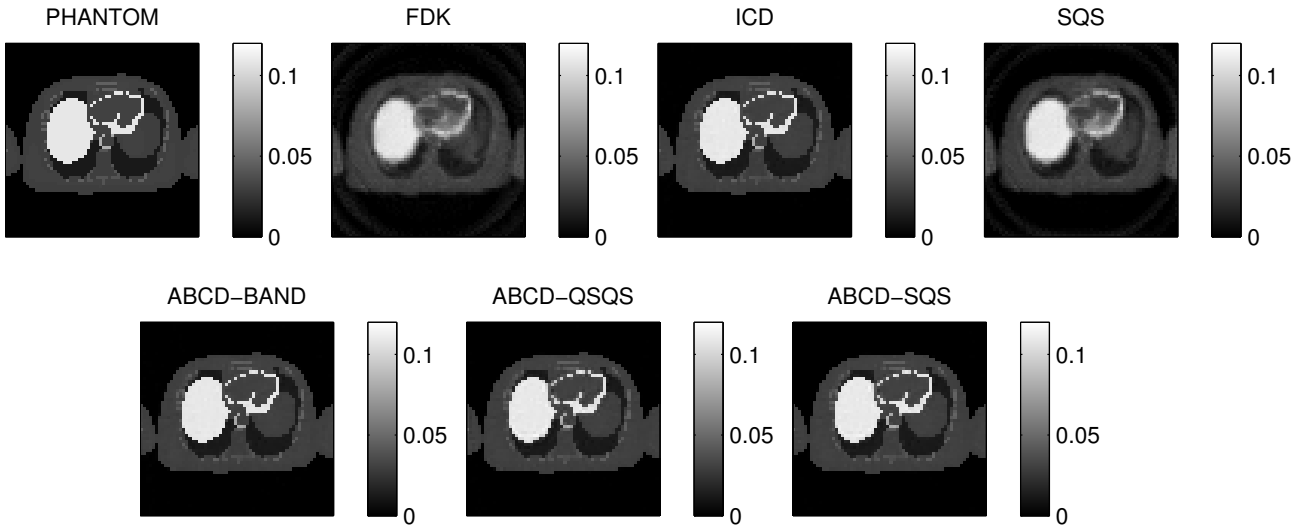


Figure 7.4: Phantom, FDK reconstructed image and reconstructed images by five different algorithms after 15 iterations

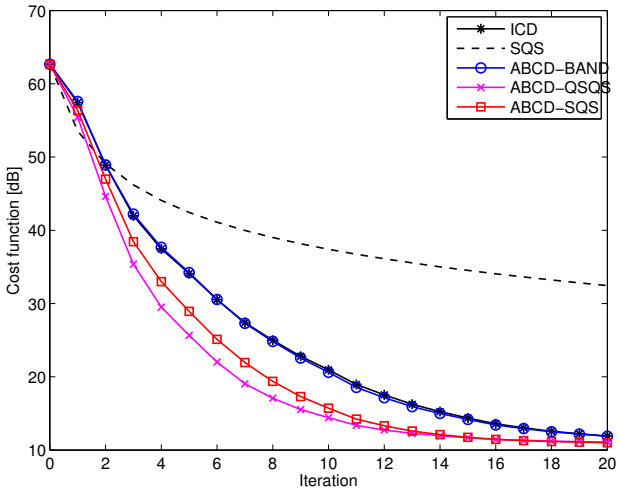


Figure 7.5: Cost function $\Psi(x^{(n)})$ versus iteration n for five algorithms

Fig. 7.4 shows one slice of the 3D phantom image, and of the reconstructed 3D image by FDK [34, 37] and by the five iterative algorithms listed in (7.12) after 15 iterations with FDK reconstructed image as the initial guess. ICD and ABCD algorithms reached nearly the same image after 15 iterations whereas the SQS reconstructed image is still close to FDK reconstructed image. However, ABCD-BAND might converge faster than ICD since

both only use quadratic surrogate approach and ABCD-BAND updates a group of pixel simultaneously. Fig. 7.5 illustrates the cost function $\Psi(x^{(n)})$ value [dB] as a function of iterations for the five algorithms. The convergence rate of SQS is slow compared with other algorithms, as expected. We conclude that the convergence rate of the four algorithms (except SQS) are similar in this simulation. We further need to plot NRMS difference value [dB] between the current image and the converged image, versus iterations, to compare the convergence rate of five algorithms more accurately.

7.3 Conclusion and Discussion

We have found that the knowledge of 3D cone-beam geometry and its projector can help developing efficent optimization algorithms for 3D X-ray CT. Based on 3D CT geometry, we have chosen an axial block for BCD algorithms that enabled both fast convergence rate and efficient computation. However, we further need to optimize the implementation of the proposed ABCD algorithm, and we leave them as a future work. We also would like to accelerate the convergence rate of ABCD algorithm using the non-homogeneous (NH) approach in coordinate descent (CD) algorithm [109], which the preliminary results are shown in our conference paper [57].

CHAPTER VIII

Conclusion and Future work

This dissertation presented various optimization algorithms for accelerated statistical 3D X-ray CT reconstruction, mainly focused on extending the state-of-the-art ordered subsets based on separable quadratic surrogates (OS-SQS) methods. Chapter III modified OS methods so that they can be stable in 3D helical cone-beam CT geometry, and suggested a sub-iteration averaging approach in OS methods for acceleration. Chapter IV proposed three novel optimization transfer approaches, and particularly, spatially nonuniform SQS (NU-SQS) encouraged larger step for the voxels that need more updates, leading to more than two-fold acceleration compared to the standard SQS in both simulated and real patient 3D CT scans. Chapter V applied momentum approach to ordered subsets (OS-momentum), providing very fast (initial) convergence rates with minimal computational overhead in our experiments with 3D CT scans. However, this OS-momentum method sometimes suffered from instability, and thus Section 5.2 introduced a relaxed scheme that prevented the algorithm from accumulating errors from the OS gradient approximation while preserving the fast convergence rate. Various relaxation schemes were examined with simulations and real patient CT data. Initial version of OS-momentum used well-known Nesterov's momentum methods. To further accelerate OS-momentum algorithms, Chapter VI proposed optimized gradient methods (OGM) that are twice as fast yet have remarkably simple implementations comparable to Nesterov's methods. The OS version of this OGM methods were found to decrease the cost function twice faster than the initial OS-momentum algorithms for a simulated CT scan. In addition to OS-type algorithms, Chapter VII suggested axial block coordinate descent (ABCD) algorithms that are specifically designed for 3D cone-beam CT geometry. The ABCD updates voxels within an axial block simultaneously and this lead to both fast convergence rate and efficient computation in 3D cone-beam CT geometry.

8.1 Future work

Among many iterative algorithms developed in this dissertation, the convergence of OS-momentum methods in Chapters V and VI was particularly fast compared to others, and we would like to further accelerate their convergence rates while improving the stability.

Continuing the development of new momentum methods that are both efficient and faster than the proposed OGM methods in Chapter VI is a natural pursuit. Considering the OS version of momentum methods, extending DT's [31] performance estimation problem (PEP) approach and our OGM methods to either stochastic or incremental gradient methods would be very interesting. In other words, investigating ways to extend our OGM methods (and DT's work [31]) to broader classes of functions, such as nonsmooth functions or strongly convex functions constrained optimization problems, or different choice of algorithms, such as stochastic or incremental gradient methods would be very important work.

The direct combination of OS and momentum can be unstable sometimes due to accumulated error, particularly for large numbers of subsets, as discussed in Chapter V. We investigated the diminishing step size rule in Section 5.2 to stabilize the algorithm while preserving the fast convergence rate. The results in Section 5.3 looked promising but the parameter tuning for the diminishing rule was found not simple, and we would like to further develop a stabilizing approach for OS-momentum algorithm with minimal additional tuning parameters.

APPENDICES

APPENDIX A

Proof of Lemma 2

The proposed choice $\tilde{\lambda}_{ij}^{(n)} = a_{ij} u_j^{(n)}$ in (4.23) and its corresponding $\tilde{d}_j^{L,(n)}$ in (4.24) are a choice that minimizes $\sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 d_j^{L,(n)}$ among all possible $d_j^{L,(n)}$ in (4.14), *i.e.*,

$$\left\{\tilde{\lambda}_{ij}^{(n)}\right\} = \underset{\{\lambda_{ij}^{(n)}\}}{\arg\min} \sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 \left(\sum_{\substack{i=1 \\ a_{ij}\neq 0}}^{N_d} \breve{c}_i^{(n)} \frac{a_{ij}^2}{\lambda_{ij}^{(n)}} \sum_{\substack{l=1 \\ a_{il}\neq 0}}^{N_p} \lambda_{il}^{(n)} \right),$$

subject to the positivity constraint on $\lambda_{ij}^{(n)}$ if $a_{ij} \neq 0$.

Proof. By the Schwarz inequality $\langle \mathbf{s}, \mathbf{t} \rangle^2 \leq ||\mathbf{s}||^2 ||\mathbf{t}||^2$, we have

$$\left(\sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{N_p} a_{ij} u_j^{(n)} \right)^2 \leq \sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{N_p} \left(u_j^{(n)}\right)^2 \frac{a_{ij}^2}{\lambda_{ij}^{(n)}} \sum_{\substack{l=1 \\ a_{il}\neq 0}}^{N_p} \lambda_{il}^{(n)},$$

where $s_j = \sqrt{\frac{\lambda_{ij}^{(n)}}{\sum_{l=1}^{N_p} \lambda_{il}^{(n)}}}$ and $t_j = a_{ij} u_j^{(n)} \sqrt{\frac{\sum_{l=1}^{N_p} \lambda_{il}^{(n)}}{\lambda_{ij}^{(n)}}}$. Then,

$$\begin{aligned}
\sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 d_j^{L,(n)} &= \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \left(\sum_{\substack{j=1 \\ a_{ij} \neq 0}}^{N_p} \left(u_j^{(n)}\right)^2 \frac{a_{ij}^2}{\lambda_{ij}^{(n)}} \sum_{\substack{l=1 \\ a_{il} \neq 0}}^{N_p} \lambda_{il}^{(n)} \right) \\
&\geq \sum_{i=1}^{N_d} \breve{c}_i^{(n)} \left(\sum_{\substack{j=1 \\ a_{ij} \neq 0}}^{N_p} a_{ij} u_j^{(n)} \right)^2 = \sum_{j=1}^{N_p} \left(\sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{N_d} \breve{c}_i^{(n)} a_{ij} u_j^{(n)} \sum_{\substack{l=1 \\ a_{il} \neq 0}}^{N_p} a_{il} u_l^{(n)} \right) \\
&= \sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 \left(\frac{1}{u_j^{(n)}} \sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{N_d} \breve{c}_i^{(n)} a_{ij} \sum_{\substack{l=1 \\ a_{il} \neq 0}}^{N_p} a_{il} u_l^{(n)} \right) = \sum_{j=1}^{N_p} \left(u_j^{(n)}\right)^2 \tilde{d}_j^{L,(n)}.
\end{aligned}$$

□

APPENDIX B

Proof of Lemma 4

We extend the proof of [29, Theorem 7] for *diagonally preconditioned* stochastic OS-SQS-type algorithms for the proof of Lemma 4. We first use the following lemma:

Lemma 9. *For $k \geq 0$, the sequence $\{x^{(\frac{k}{M})}\}$ generated by Table 5.3 satisfies*

$$\sum_{l=0}^{k} t_l \Psi(x^{(\frac{k+1}{M})}) - e^{(k)} \leq \min_{v \succeq 0} \Phi^{(k)}(v)$$
$$\Phi^{(k)}(v) \leq \sum_{l=0}^{k} t_l \Psi(v) + \frac{||v - z^{(0)}||^2_{\Gamma^{(k)}}}{2} + \bar{e}^{(k)}(v),$$

where

$$\Phi^{(k)}(v) \triangleq \sum_{l=0}^{k} t_l \left[M\Psi_{S_l}(z^{(\frac{l}{M})}) + M\nabla\Psi_{S_l}(z^{(\frac{l}{M})})' \left(v - z^{(\frac{l}{M})} \right) \right] + \frac{||v - z^{(0)}||^2_{\Gamma^{(k)}}}{2}$$

that satisfies $v^{(\frac{k+1}{M})} = \arg\min_{v \succeq 0} \Phi^{(k)}(v)$,

$$e^{(k)} \triangleq \sum_{l=0}^{k} \sum_{i=0}^{l} t_i \left\| M\nabla\Psi_{S_l}(z^{(\frac{l}{M})}) - \nabla\Psi(z^{(\frac{l}{M})}) \right\|^2_{\left[\Gamma^{(l)} - D\right]^{-1}}$$
$$+ \sum_{l=0}^{k} t_l \left(\Psi(z^{(\frac{l}{M})}) - M\Psi_{S_l}(z^{(\frac{l}{M})}) \right) + \sum_{l=1}^{k} \sum_{i=0}^{l-1} t_i \left(\nabla\Psi(z^{(\frac{l}{M})}) - M\nabla\Psi_{S_l}(z^{(\frac{l}{M})}) \right)' \left(z^{(\frac{l}{M})} - x^{(\frac{l}{M})} \right)$$

and

$$\bar{e}^{(k)}(v) \triangleq \sum_{l=0}^{k} t_l \left[M\Psi_{S_l}(z^{(\frac{l}{M})}) - \Psi(z^{(\frac{l}{M})}) + \left(M\nabla\Psi_{S_l}(z^{(\frac{l}{M})}) - \nabla\Psi(z^{(\frac{l}{M})}) \right)' \left(v - z^{(\frac{l}{M})} \right) \right].$$

Proof. Simply generalize the proof of [29, Lemma 2] using the proof of [83, Lemma 1]. □

Using Lemma 9 with the fact $\min_{v \succeq 0} \Phi^{(k)}(v) \leq \Phi^{(k)}(\hat{x})$ leads to the following:

$$\sum_{l=0}^{k} t_l \left(\Psi(x^{(\frac{k+1}{M})}) - \Psi(\hat{x}) \right) \leq \frac{||x^{(0)} - \hat{x}||^2_{\Gamma^{(k)}}}{2} + e^{(k)} + \bar{e}^{(k)}(\hat{x}).$$

Finally, the expectation on the above equation provides Lemma 4, as in [29, Theorem 7].

APPENDIX C

Choice of coefficients t_k

Lemma 10. *For any given* $\{\gamma_j^{(k)}\}$ *satisfying its constraint in line* 5 *of Table 5.3, the* $\{t_k\}$ *generated by* $t_0 = 1$

$$t_{k+1} = \frac{1}{2\alpha^{(k+1)}}\left(1 + \sqrt{1 + 4t_k^2\alpha^{(k)}\alpha^{(k+1)}}\right),$$

where $\alpha^{(k+1)} \triangleq \max_j \left(\gamma_j^{(k+1)}/\gamma_j^{(k)}\right)$ *and* $\alpha^{(0)} = 1$*, tightly satisfies the following conditions:*

$$t_0 \in (0,\ 1] \text{ and } \alpha^{(k+1)}t_{k+1}^2 \leq \sum_{l=0}^{k+1} t_l, \forall k \geq 0,$$

which are equivalent to the conditions in line 6 of Table 5.3.

Proof. Let t_0 have the largest possible value 1.

For $k = 0$,

$$\begin{aligned} &\alpha^{(1)}t_1^2 = t_0 + t_1 = t_0^2 + t_1 \\ &t_1 = \frac{1}{2\alpha^{(1)}}\left(1 + \sqrt{1 + 4t_0^2\alpha^{(0)}\alpha^{(1)}}\right). \end{aligned} \tag{C.1}$$

For $k > 0$, we get

$$\alpha^{(k+1)}t_{k+1}^2 = \sum_{l=0}^{k+1} t_l = \alpha^{(k)}t_k^2 + t_{k+1}$$

$$t_{k+1} = \frac{1}{2\alpha^{(k+1)}} \left(1 + \sqrt{1 + 4t_k^2 \alpha^{(k)} \alpha^{(k+1)}} \right). \tag{C.2}$$

□

This rule for t_k (C.2) reduces to those used in Tables 5.1 and 5.2 when $\gamma_j^{(k+1)} = \gamma_j^{(k)}$ for all $k \geq 0$ and j.

APPENDIX D

Proof of Lemma 6

We prove that the choice $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \gamma)$ in (6.30), (6.31), (6.32) and (6.33) satisfies the feasible conditions (6.35) of (RD1).

Using the definition of $\breve{\boldsymbol{Q}}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau})$ in (6.28), and considering the first two conditions of (6.35), we get

$$\begin{aligned}\lambda_{i+1} = \breve{Q}_{i,i}(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = 2\breve{q}_i^2(\boldsymbol{r}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = \frac{1}{2\tau_N^2}\tau_i^2 \\ = \begin{cases} \frac{1}{2(1-\lambda_N)^2}\lambda_1^2, & i = 0 \\ \frac{1}{2(1-\lambda_N)^2}(\lambda_{i+1} - \lambda_i)^2, & i = 1, \cdots, N-1, \end{cases}\end{aligned}$$

where the last equality comes from $(\boldsymbol{\lambda}, \boldsymbol{\tau}) \in \Lambda$, and this reduces to the following recursion:

$$\begin{cases} \lambda_1 = 2(1-\lambda_N)^2, \\ (\lambda_i - \lambda_{i-1})^2 - \lambda_1\lambda_i = 0. \quad i = 2, \cdots, N. \end{cases} \tag{D.1}$$

We use induction to prove that the solution of (D.1) is

$$\lambda_i = \begin{cases} \frac{2}{\theta_N^2}, & i = 1, \\ \theta_{i-1}^2\lambda_1, & i = 2, \cdots, N, \end{cases}$$

which is equivalent to $\hat{\boldsymbol{\lambda}}$ (6.31). It is obvious that $\lambda_1 = \theta_0\lambda_1$, and for $i = 2$ in (D.1), we get

$$\lambda_2 = \frac{3\lambda_1 + \sqrt{9\lambda_1^2 - 4\lambda_1^2}}{2} = \frac{3+\sqrt{5}}{2}\lambda_1 = \theta_1^2\lambda_1.$$

Then, assuming $\lambda_i = \theta_{i-1}^2 \lambda_1$ for $i = 1, \cdots, n$ and $n \leq N-1$, and using the second equality in (D.1) for $i = n+1$, we get

$$\begin{aligned}\lambda_{n+1} &= \frac{\lambda_1 + 2\lambda_n + \sqrt{(\lambda_1 + 2\lambda_n)^2 - 4\lambda_n^2}}{2} = \frac{1 + 2\theta_{n-1}^2 + \sqrt{1 + 4\theta_{n-1}^2}}{2}\lambda_1 \\ &= \left(\theta_{n-1}^2 + \frac{1 + \sqrt{1 + 4\theta_{n-1}^2}}{2}\right)\lambda_1 = \theta_n^2 \lambda_1,\end{aligned}$$

where the last equality uses (6.5). Then we use the first equality in (D.1) to find the value of λ_1 as

$$\begin{aligned}&\lambda_1 = 2(1 - \theta_{N-1}^2 \lambda_1)^2 \\ &\theta_{N-1}^4 \lambda_1^2 - 2(\theta_{N-1}^2 + \frac{1}{4})\lambda_1 + \frac{1}{2} = 0 \\ &\lambda_1 = \frac{\theta_{N-1}^2 + \frac{1}{4} - \sqrt{(\theta_{N-1}^2 + \frac{1}{4})^2 - \theta_{N-1}^4}}{\theta_{N-1}^4} = \frac{1}{\theta_{N-1}^2 + \frac{1}{4} + \sqrt{\frac{\theta_{N-1}^2}{2} + \frac{1}{16}}} \\ &\quad = \frac{8}{\left(1 + \sqrt{1 + 8\theta_{N-1}^2}\right)^2} = \frac{2}{\theta_N^2}\end{aligned}$$

with θ_N in (6.34).

Until now, we derived $\hat{\boldsymbol{\lambda}}$ (6.31) using some conditions of (6.35). Consequently, using the last two conditions in (6.35) with (6.5) and (6.36), we can easily derive the following:

$$\begin{aligned}\tau_i &= \begin{cases} \hat{\lambda}_1 = \frac{2}{\theta_N^2}, & i = 0, \\ \hat{\lambda}_{i+1} - \hat{\lambda}_i = \frac{2\theta_i^2}{\theta_N^2} - \frac{2\theta_{i-1}^2}{\theta_N^2} = \frac{2\theta_i}{\theta_N^2}, & i = 1, \cdots, N-1, \\ 1 - \hat{\lambda}_N = 1 - \frac{2\theta_{N-1}^2}{\theta_N^2} = \frac{1}{\theta_N}, & i = N, \end{cases} \\ \gamma &= \tau_N^2 = \frac{1}{\theta_N^2},\end{aligned}$$

which are equivalent to $\hat{\boldsymbol{\tau}}$ (6.32) and $\hat{\gamma}$ (6.33).

Next, we derive $\hat{\boldsymbol{r}}$ for given $\hat{\boldsymbol{\lambda}}$ (6.31) and $\hat{\boldsymbol{\tau}}$ (6.32). Inserting $\hat{\boldsymbol{\tau}}$ (6.32) to the first two conditions of (6.35), we get

$$\begin{cases} \breve{q}_i(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}) = \frac{\hat{\tau}_i}{2\hat{\tau}_N} = \frac{\theta_i}{\theta_N}, \\ \breve{Q}_{i,k}(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}) = 2\breve{q}_i(\boldsymbol{r}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}})\breve{q}_k(\boldsymbol{r}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}) = \frac{2\theta_i\theta_k}{\theta_N^2}, \end{cases} \tag{D.2}$$

for $i,k=0,\cdots,N-1$, and considering (6.26) and (D.2), we get

$$\breve{S}_{i,k}(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}})=\begin{cases}\frac{2\theta_i\theta_k}{\theta_N^2}, & i,k=0,\cdots,N-1,\\ \frac{\theta_i}{\theta_N} & i=0,\cdots,N-1,\ k=N,\\ \frac{\theta_k}{\theta_N}, & i=N,\ k=0,\cdots,N-1,\\ \frac{1}{2} & i=N,\ k=N.\end{cases} \tag{D.3}$$

Finally, using the two equivalent forms (6.23) and (D.3) of $\breve{\boldsymbol{S}}(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}})$, we get

$$\breve{S}_{i,k}(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}})\begin{cases}\frac{1}{2}\hat{r}_{i,k}=\frac{2\theta_i\theta_k}{\theta_N^2}, & i=2,\cdots,N-1,\ k=0,\cdots,i-2,\\ \frac{1}{2}(\hat{r}_{i,k}-\hat{\lambda}_i)=\frac{2\theta_i\theta_k}{\theta_N^2}, & i=1,\cdots,N-1,\ k=i-1,\\ \frac{1}{2}\hat{r}_{i,k}=\frac{\theta_k}{\theta_N}, & i=N,\ k=0,\cdots,i-2,\\ \frac{1}{2}(\hat{r}_{i,k}-\hat{\lambda}_i)=\frac{\theta_k}{\theta_N}. & i=1,\cdots,N-1,\ k=i-1,\end{cases} \tag{D.4}$$

and this can be easily converted to the choice $\hat{r}_{i,k}$ in (6.30).

For these given $(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}})$, we can easily notice that

$$\begin{pmatrix}\breve{\boldsymbol{S}}(\hat{\boldsymbol{h}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}}) & \frac{1}{2}\hat{\boldsymbol{\tau}}\\ \frac{1}{2}\hat{\boldsymbol{\tau}}^T & \frac{1}{2}\hat{\gamma}\end{pmatrix}=\begin{pmatrix}\frac{2}{\theta_N^2}\breve{\boldsymbol{\theta}}\breve{\boldsymbol{\theta}}^T & \frac{1}{\theta_N^2}\breve{\boldsymbol{\theta}}\\ \frac{1}{\theta_N^2}\breve{\boldsymbol{\theta}}^T & \frac{1}{2\theta_N^2}\end{pmatrix}=\frac{2}{\theta_N^2}\begin{pmatrix}\breve{\boldsymbol{\theta}}\\ \frac{1}{2}\end{pmatrix}\begin{pmatrix}\breve{\boldsymbol{\theta}}\\ \frac{1}{2}\end{pmatrix}^T\succeq 0 \tag{D.5}$$

for $\breve{\boldsymbol{\theta}}=\big(\theta_0,\cdots,\theta_{N-1},\frac{\theta_N}{2}\big)^T$, showing that the choice is feasible in both (RD) and (RD1).

APPENDIX E

Proof of Lemma 7

We prove that the choice $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \gamma)$ in (6.30), (6.31), (6.32) and (6.33) is a solution of problem (RD) using KKT conditions.

We first rewrite problem (RD) into a general form:

$$\arg\min_{\boldsymbol{r}\in\mathbb{R}^{N(N+1)/2}} \min_{\substack{\boldsymbol{\lambda}\in\mathbb{R}^N,\\ \boldsymbol{\tau}\in\mathbb{R}^{N+1},\\ \gamma\in\mathbb{R}}} \left\{ \frac{1}{2}LR^2\gamma \ :\ \boldsymbol{F}_0(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma) \succeq 0,\ \boldsymbol{F}_1(\boldsymbol{\lambda},\boldsymbol{\tau}) \geq 0,\ \boldsymbol{F}_2(\boldsymbol{\lambda},\boldsymbol{\tau}) = 0 \right\}, \quad \text{(RD2)}$$

where

$$\begin{cases} \boldsymbol{F}_0(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma) = \begin{pmatrix} \breve{\boldsymbol{S}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau}) & \frac{1}{2}\boldsymbol{\tau} \\ \frac{1}{2}\boldsymbol{\tau}^T & \frac{1}{2}\gamma \end{pmatrix} \in \mathbb{R}^{(N+2)\times(N+2)}, \\ \boldsymbol{F}_1(\boldsymbol{\lambda},\boldsymbol{\tau}) = (\lambda_1,\cdots,\lambda_N,\tau_0,\cdots,\tau_N)^T \in \mathbb{R}^{2N+1}, \\ \boldsymbol{F}_2(\boldsymbol{\lambda},\boldsymbol{\tau}) = (\tau_0-\lambda_1, \lambda_1-\lambda_2+\tau_1, \cdots, \lambda_{N-1}-\lambda_N+\tau_{N-1}, \lambda_N+\tau_N-1)^T \in \mathbb{R}^{N+1}. \end{cases}$$

The Lagrangian of problem (RD2) is

$$\breve{\mathcal{L}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma,\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu}) = \frac{1}{2}LR^2\gamma - \mathsf{tr}\{\boldsymbol{F}_0(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma)\boldsymbol{Z}\} - \boldsymbol{\eta}^T\boldsymbol{F}_1(\boldsymbol{\lambda},\boldsymbol{\tau}) - \boldsymbol{\mu}^T\boldsymbol{F}_2(\boldsymbol{\lambda},\boldsymbol{\tau}), \quad \text{(E.1)}$$

and the KKT conditions of (RD2) are

$$\begin{cases} \boldsymbol{F}_0(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma) \succeq 0,\ \boldsymbol{F}_1(\boldsymbol{\lambda},\boldsymbol{\tau}) \geq 0,\ \boldsymbol{F}_2(\boldsymbol{\lambda},\boldsymbol{\tau}) = 0, \\ \nabla_{\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma}\breve{\mathcal{L}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma,\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu}) = 0, \\ \boldsymbol{Z} \succeq 0,\ \boldsymbol{\eta} \geq 0, \\ \mathsf{tr}\{\boldsymbol{F}_0(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma)\boldsymbol{Z}\} = 0,\ \boldsymbol{\eta}^T\boldsymbol{F}_1(\boldsymbol{\lambda},\boldsymbol{\tau}) = 0, \end{cases} \quad \text{(E.2)}$$

where $\boldsymbol{Z}$ is a $(N+2)\times(N+2)$ symmetric matrix, $\boldsymbol{\eta}\in\mathbb{R}^{2N+1}$ and $\boldsymbol{\mu}\in\mathbb{R}^{N+1}$. Since $\frac{1}{2}LR^2\gamma$ is linear, a pair $(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma)$ satisfying the KKT conditions (E.2) with a pair $(\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu})$ becomes an optimal solution of (RD2).

Hereafter, we show that a pair $(\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu})$ satisfying the KKT conditions (E.2) for a given feasible $(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})$ in Lemma 7 exists. The choice $(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})$ is feasible, so the first condition of (E.2) is satisfied. We can easily get $\boldsymbol{\eta}=0$ from the last two conditions of (E.2), since $\boldsymbol{F}_1(\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}})>0$. In addition, for symmetric matrices $\boldsymbol{F}_0(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})\succeq 0$ and $Z\succeq 0$, we can replace the condition $\mathsf{tr}\Big\{\boldsymbol{F}_0(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})Z\Big\}=0$ by $\boldsymbol{F}_0(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})Z=0$. The stationary condition $\nabla_{\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma}\breve{\mathcal{L}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma,\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu})=0$ of (E.2) can be rewritten as

$$\begin{cases}\frac{\partial}{\partial r_{i,k}}\breve{\mathcal{L}}=-\frac{1}{2}(Z_{i,k}+Z_{k,i})=0, & i=1,\cdots,N,\ k=0,\cdots,i-1,\\ \frac{\partial}{\partial \lambda_i}\breve{\mathcal{L}}=-\frac{1}{2}(Z_{i-1,i-1}-Z_{i-1,i}-Z_{i,i-1}+Z_{i,i})-(\mu_i-\mu_{i-1})=0, & i=1,\cdots,N,\\ \frac{\partial}{\partial \tau_i}\breve{\mathcal{L}}=-\frac{1}{2}(Z_{i,i}+Z_{i,N+1}+Z_{N+1,i})-\mu_i=0, & i=0,\cdots,N,\\ \frac{\partial}{\partial \gamma}\breve{\mathcal{L}}=\frac{1}{2}LR^2-\frac{1}{2}Z_{N+1,N+1}=0,\end{cases}$$

where we omit the arguments $(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma,\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu})$ in $\breve{\mathcal{L}}(\boldsymbol{r},\boldsymbol{\lambda},\boldsymbol{\tau},\gamma,\boldsymbol{Z},\boldsymbol{\eta},\boldsymbol{\mu})$ for notational simplicity.

Then the KKT conditions (E.2) for given $(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})$ reduces to

$$\begin{cases}Z_{i,k}=0, & i=1,\cdots,N,\ k=0,\cdots,i-1,\\ Z_{i-1,i-1}+Z_{i,i}=-2(\mu_i-\mu_{i-1}), & i=1,\cdots,N\\ Z_{i,i}+2Z_{N+1,i}=-2\mu_i, & i=0,\cdots,N\\ Z_{N+1,N+1}=LR^2,\\ \boldsymbol{F}_0(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})\boldsymbol{Z}=0,\\ \boldsymbol{Z}\succeq 0,\end{cases}\tag{E.3}$$

and a careful reformulation of the condition $\boldsymbol{F}_0(\hat{\boldsymbol{r}},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\tau}},\hat{\gamma})\boldsymbol{Z}=0$ leads to

$$\begin{cases}Z_{N+1,i}=-2\theta_i Z_{i,i}, & i=0,\cdots,N-1,\\ Z_{N+1,N}=-\theta_N Z_{N,N},\\ \sum_{i=0}^{N-1}2\theta_i Z_{N+1,i}+\theta_N Z_{N+1,N}+LR^2=0.\end{cases}\tag{E.4}$$

Inserting (E.4) into (E.3) with additional derivation, the conditions (E.3) can be reformulated

as

$$\boldsymbol{Z} \succeq 0, \quad \sum_{i=0}^{N-1} 2\theta_i Z_{N+1,i} + \theta_N Z_{N+1,N} + LR^2 = 0,$$

$$Z_{i,k} = \begin{cases} \frac{2\theta_{i-1}-1}{2\theta_i} Z_{i-1,i-1}, & i = 1, \cdots, N-1,\ k = i, \\ \frac{2\theta_{i-1}-1}{\theta_i} Z_{i-1,i-1}, & i = N,\ k = i, \\ -2\theta_i Z_{i,i}, & i = N+1,\ k = 0, \cdots, i-2, \\ -\theta_i Z_{i,i}, & i = N+1,\ k = i-1, \\ LR^2, & i = N+1,\ k = i, \\ 0, & i = 1, \cdots, N,\ k = 0, \cdots, i-1, \end{cases}$$

$$\mu_i = \begin{cases} \frac{4\theta_i - 1}{2} Z_{i,i}, & i = 0, \cdots, N-1, \\ \frac{2\theta_i - 1}{2} Z_{N,N}, & i = N. \end{cases}$$

From the above equalities, we can write all variables with respect to $Z_{0,0}$ as

$$Z_{i,k} = \begin{cases} LR^2 \left(\sum_{i=0}^{N-1} 4\theta_i^2 \kappa_i + 2\theta_N^2 \kappa_N \right)^{-1}, & i = 0,\ k = i, \\ \kappa_i Z_{0,0}, & i = 1, \cdots, N-1,\ k = i, \\ 2\kappa_i Z_{0,0}, & i = N,\ k = i, \\ -2\theta_k \kappa_k Z_{0,0}, & i = N+1,\ k = 0, \cdots, i-1, \\ LR^2, & i = N+1,\ k = i, \\ 0, & i = 1, \cdots, N,\ k = 0, \cdots, i-1, \end{cases}$$

$$\mu_i = \begin{cases} \frac{4\theta_i - 1}{2} \kappa_i Z_{0,0}, & i = 0, \cdots, N-1, \\ (2\theta_i - 1)\kappa_i Z_{0,0}, & i = N, \end{cases}$$

where $\kappa_i = \prod_{j=1}^{i} \frac{2\theta_{j-1}-1}{2\theta_j}$.

We conclude by showing that the matrix $\boldsymbol{Z}$ is positive semidefinite:

$$\begin{aligned} \boldsymbol{Z} = & \left(\sum_{i=0}^{N-1} \kappa_i (\hat{\boldsymbol{u}}_i - 2\theta_i \hat{\boldsymbol{u}}_{N+1})(\hat{\boldsymbol{u}}_i - 2\theta_i \hat{\boldsymbol{u}}_{N+1})^T \right. \\ & \left. + 2\kappa_N (\hat{\boldsymbol{u}}_N - \theta_N \hat{\boldsymbol{u}}_{N+1})(\hat{\boldsymbol{u}}_N - \theta_N \hat{\boldsymbol{u}}_{N+1})^T \right) Z_{0,0} \succeq 0, \end{aligned}$$

for $\hat{\boldsymbol{u}}_i = \boldsymbol{e}_{N+2,i+1} \in \mathbb{R}^{N+2}$. This finally verifies the existence of $(\boldsymbol{Z}, \boldsymbol{\eta}, \boldsymbol{\mu})$ that satisfies the

KKT conditions (E.2) for a given $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\tau}}, \hat{\gamma})$.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] H. Erdoğan and J. A. Fessler, "Monotonic algorithms for transmission tomography," *IEEE Trans. Med. Imag.*, vol. 18, no. 9, pp. 801–14, Sep. 1999.

[2] ——, "Ordered subsets algorithms for transmission tomography," *Phys. Med. Biol.*, vol. 44, no. 11, pp. 2835–51, Nov. 1999.

[3] P. J. La Rivière, J. Bian, and P. A. Vargas, "Penalized-likelihood sinogram restoration for computed tomography," *IEEE Trans. Med. Imag.*, vol. 25, no. 8, pp. 1022–36, Aug. 2006.

[4] S. Ahn and J. A. Fessler, "Globally convergent image reconstruction for emission tomography using relaxed ordered subsets algorithms," *IEEE Trans. Med. Imag.*, vol. 22, no. 5, pp. 613–26, May 2003.

[5] S. Ahn, J. A. Fessler, D. Blatt, and A. O. Hero, "Convergent incremental optimization transfer algorithms: Application to tomography," *IEEE Trans. Med. Imag.*, vol. 25, no. 3, pp. 283–96, Mar. 2006.

[6] G. I. Angelis, A. J. Reader, F. A. Kotasidis, W. R. Lionheart, and J. C. Matthews, "The performance of monotonic and new non-monotonic gradient ascent reconstruction algorithms for high-resolution neuroreceptor PET imaging," *Phys. Med. Biol.*, vol. 56, no. 13, pp. 3895–917, Jul. 2011.

[7] S. Anthoine, J.-F. Aujol, Y. Boursier, and C. Mélot, "Some proximal methods for Poisson intensity CBCT and PET," *Inverse Prob. and Imaging*, vol. 6, no. 4, pp. 565–98, Nov. 2012.

[8] A. Beck, "Quadratic matrix programming," *SIAM J. Optim.*, vol. 17, no. 4, pp. 1224–38, 2006.

[9] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Im. Proc.*, vol. 18, no. 11, pp. 2419–34, Nov. 2009.

[10] ——, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[11] A. Ben-Tal and A. Nemirovski, *Lecture on modern convex optimization: Analysis, algorithms, and engineering applications*. Philadelphia: Soc. Indust. Appl. Math., 2001.

[12] T. M. Benson, B. K. B. D. Man, L. Fu, and J.-B. Thibault, "Block-based iterative coordinate descent," in *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, 2010, pp. 2856–9.

[13] D. P. Bertsekas, "Multiplier methods: A survey," *Automatica*, vol. 12, no. 2, pp. 133–45, Mar. 1976.

[14] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Trans. Im. Proc.*, vol. 5, no. 3, pp. 480–92, Mar. 1996.

[15] D. J. Brenner and E. J. Hall, "Computed tomography- An increasing source of radiation exposure," *New England journal of Medicine*, vol. 357, no. 22, pp. 2277–84, Nov. 2007.

[16] K. M. Brown, T. Köhler, F. Bergner, R. Bippus, B. Brendel, S. Zabic, W. C. Karl, S. Singh, A. Padole, and S. Do, "Sparse sampling for CT dose reduction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 428–31.

[17] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proc. IEEE Intl. Conf. on Image Processing*, vol. 2, 1994, pp. 168–71.

[18] J. H. Cho and J. A. Fessler, "Accelerating ordered-subsets image reconstruction for X-ray CT using double surrogates," in *Proc. SPIE 8313 Medical Imaging 2012: Phys. Med. Im.*, 2012, p. 83131X.

[19] K. Choi, J. Wang, L. Zhu, T.-S. Suh, S. Boyd, and L. Xing, "Compressed sensing based cone-beam computed tomography reconstruction with a first-order method," *Med. Phys.*, vol. 37, no. 9, pp. 5113–25, Nov. 2010.

[20] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–57, Nov. 2004.

[21] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Phys. Med. Biol.*, vol. 49, no. 11, pp. 2463–75, Jun. 2004.

[22] B. De Man, S. Basu, J.-B. Thibault, J. Hsieh, J. A. Fessler, C. Bouman, and K. Sauer, "A study of different minimization approaches for iterative reconstruction in X-ray CT," in *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, vol. 5, 2005, pp. 2708–10.

[23] A. R. De Pierro, "On the convergence of the iterative image space reconstruction algorithm for volume ECT," *IEEE Trans. Med. Imag.*, vol. 6, no. 2, pp. 174–5, Jun. 1987.

[24] ——, "On the relation between the ISRA and the EM algorithm for positron emission tomography," *IEEE Trans. Med. Imag.*, vol. 12, no. 2, pp. 328–33, Jun. 1993.

[25] ——, "A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography," *IEEE Trans. Med. Imag.*, vol. 14, no. 1, pp. 132–7, Mar. 1995.

[26] M. Defrise, F. Noo, and H. Kudo, "A solution to the long-object problem in helical cone-beam tomography," *Phys. Med. Biol.*, vol. 45, no. 3, pp. 623–43, Mar. 2000.

[27] A. H. Delaney and Y. Bresler, "Globally convergent edge-preserving regularized reconstruction: an application to limited-angle tomography," *IEEE Trans. Im. Proc.*, vol. 7, no. 2, pp. 204–21, Feb. 1998.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Stat. Soc. Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[29] O. Devolder, "Stochastic first order methods in smooth convex optimization," 2011.

[30] O. Devolder, François. Glineur, and Y. Nesterov, "Intermediate gradient methods for smooth convex problems with inexact oracle," 2013, cORE discussion paper 2013/17.

[31] Y. Drori and M. Teboulle, "Performance of first-order methods for smooth convex minimization: A novel approach," *Mathematical Programming*, 2013.

[32] I. A. Elbakri and J. A. Fessler, "Statistical image reconstruction for polyenergetic X-ray computed tomography," *IEEE Trans. Med. Imag.*, vol. 21, no. 2, pp. 89–99, Feb. 2002.

[33] R. C. Fair, "On the robust estimation of econometric models," *Ann. Econ. Social Measurement*, vol. 2, pp. 667–77, Oct. 1974.

[34] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone beam algorithm," *J. Opt. Soc. Am. A*, vol. 1, no. 6, pp. 612–9, Jun. 1984.

[35] J. A. Fessler, "Hybrid Poisson/polynomial objective functions for tomographic image reconstruction from transmission scans," *IEEE Trans. Im. Proc.*, vol. 4, no. 10, pp. 1439–50, Oct. 1995.

[36] ——, "Statistical image reconstruction methods for transmission tomography," in *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*, M. Sonka and J. M. Fitzpatrick, Eds. Bellingham: SPIE, 2000, pp. 1–70.

[37] ——, "Matlab tomography toolbox," 2004, available from `http://www.eecs.umich.edu/~fessler`.

[38] J. A. Fessler and S. D. Booth, "Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction," *IEEE Trans. Im. Proc.*, vol. 8, no. 5, pp. 688–99, May 1999.

[39] J. A. Fessler, N. H. Clinthorne, and W. L. Rogers, "On complete data spaces for PET reconstruction algorithms," *IEEE Trans. Nuc. Sci.*, vol. 40, no. 4, pp. 1055–61, Aug. 1993.

[40] J. A. Fessler, E. P. Ficaro, N. H. Clinthorne, and K. Lange, "Fast parallelizable algorithms for transmission image reconstruction," in *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, vol. 3, 1995, pp. 1346–50.

[41] ——, "Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction," *IEEE Trans. Med. Imag.*, vol. 16, no. 2, pp. 166–75, Apr. 1997.

[42] J. A. Fessler and D. Kim, "Axial block coordinate descent (ABCD) algorithm for X-ray CT image reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2011, pp. 262–5.

[43] J. A. Fessler and W. L. Rogers, "Spatial resolution properties of penalized-likelihood image reconstruction methods: Space-invariant tomographs," *IEEE Trans. Im. Proc.*, vol. 5, no. 9, pp. 1346–58, Sep. 1996.

[44] L. Fu, Z. Yu, J.-B. Thibault, B. D. Man, M. G. McGaffin, and J. A. Fessler, "Space-variant channelized preconditioner design for 3D iterative CT reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 205–8.

[45] L. Fu, K. Zeng, T. M. Benson, B. De Man, Z. Yu, G. Cao, and J.-B. Thibault, "A preliminary investigation of 3D preconditioned conjugate gradient reconstruction for cone-beam CT," in *Proc. SPIE 8313 Medical Imaging 2012: Phys. Med. Im.*, 2012, p. 83133O.

[46] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imaging Sci.*, vol. 2, no. 2, pp. 323–43, 2009.

[47] G. H. Golub and C. F. Van Loan, *Matrix computations*, 2nd ed. Johns Hopkins Univ. Press, 1989.

[48] M. Grant, S. Boyd, and Y. Ye, "Disciplined convex programming," 2006.

[49] R. J. Hathaway and J. C. Bezdek, "Grouped coordinate minimization using Newton's method for inexact minimization in one vector coordinate," *J. Optim. Theory Appl.*, vol. 71, no. 3, pp. 503–16, Dec. 1991.

[50] G. T. Herman and L. B. Meyer, "Algebraic reconstruction techniques can be made computationally efficient," *IEEE Trans. Med. Imag.*, vol. 12, no. 3, pp. 600–9, Sep. 1993.

[51] P. J. Huber, *Robust statistics*. New York: Wiley, 1981.

[52] H. M. Hudson and R. S. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *IEEE Trans. Med. Imag.*, vol. 13, no. 4, pp. 601–9, Dec. 1994.

[53] M. W. Jacobson and J. A. Fessler, "An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms," *IEEE Trans. Im. Proc.*, vol. 16, no. 10, pp. 2411–22, Oct. 2007.

[54] S. T. Jensen, S. Johansen, and S. L. Lauritzen, "Globally convergent algorithms for maximizing a likelihood function," *Biometrika*, vol. 78, no. 4, pp. 867–77, Dec. 1991.

[55] T. L. Jensen, J. H. Jrgensen, P. C. Hansen, and S. H. Jense, "Implementation of an optimal first-order method for strongly convex total variation regularization," *BIT Numerical Mathematics*, vol. 52, no. 2, pp. 329–56, Jun. 2012.

[56] D. Kim and J. A. Fessler, "Accelerated ordered-subsets algorithm based on separable quadratic surrogates for regularized image reconstruction in X-ray CT," in *Proc. IEEE Intl. Symp. Biomed. Imag.*, 2011, pp. 1134–7.

[57] ——, "Parallelizable algorithms for X-ray CT image reconstruction with spatially non-uniform updates," in *Proc. 2nd Intl. Mtg. on image formation in X-ray CT*, 2012, pp. 33–6.

[58] ——, "Ordered subsets acceleration using relaxed momentum for X-ray CT image reconstruction," in *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.*, 2013, to appear.

[59] ——, "Optimized first-order methods for smooth convex minimization," 2014, in preparation.

[60] ——, "Optimized momentum steps for accelerating X-ray CT ordered subsets image reconstruction," in *Proc. 3rd Intl. Mtg. on image formation in X-ray CT*, 2014, to appear.

[61] D. Kim, D. Pal, J.-B. Thibault, and J. A. Fessler, "Improved ordered subsets algorithm for 3D X-ray CT image reconstruction," in *Proc. 2nd Intl. Mtg. on image formation in X-ray CT*, 2012, pp. 378–81.

[62] ——, "Accelerating ordered subsets image reconstruction for X-ray CT using spatially non-uniform optimization transfer," *IEEE Trans. Med. Imag.*, vol. 32, no. 11, pp. 1965–78, Nov. 2013.

[63] D. Kim, S. Ramani, and J. A. Fessler, "Accelerating X-ray CT ordered subsets image reconstruction with Nesterov's first-order methods," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 22–5.

[64] ——, "Ordered subsets with momentum for accelerated X-ray CT image reconstruction," in *Proc. IEEE Conf. Acoust. Speech Sig. Proc.*, 2013, pp. 920–3.

[65] ——, "Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction," *IEEE Trans. Med. Imag.*, 2014, submitted.

[66] K. Lange and J. A. Fessler, "Globally convergent algorithms for maximum a posteriori transmission tomography," *IEEE Trans. Im. Proc.*, vol. 4, no. 10, pp. 1430–8, Oct. 1995.

[67] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *J. Computational and Graphical Stat.*, vol. 9, no. 1, pp. 1–20, Mar. 2000.

[68] S.-J. Lee, "Accelerated coordinate descent methods for bayesian reconstruction using ordered subsets of projection data," in *Proc. SPIE 4121 Mathematical Modeling, Estimation, and Imaging*, 2000, pp. 170–81.

[69] Y. Long, L. Cheng, X. Rui, B. De Man, A. M. Alessio, E. Asma, and P. E. Kinahan, "Analysis of ultra-low dose CT acquisition protocol and reconstruction algorithm combinations for PET attenuation correction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 400–3.

[70] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE Trans. Med. Imag.*, vol. 29, no. 11, pp. 1839–50, Nov. 2010.

[71] Z. Q. Luo, "On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks," *Neural Computation*, vol. 32, no. 2, pp. 226–45, Jun. 1991.

[72] O. L. Mangasarian and M. V. Solodov, "Serial and parallel backpropagation convergence via nonmonotone perturbed minimization," *Optimization Methods and Software*, vol. 4, no. 2, pp. 103–16, 1994.

[73] M. McGaffin and J. A. Fessler, "Sparse shift-varying FIR preconditioners for fast volume denoising," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 284–7.

[74] M. G. McGaffin and J. A. Fessler, "Duality-based projection-domain tomography solver for splitting-based X-ray CT reconstruction," in *Proc. 3rd Intl. Mtg. on image formation in X-ray CT*, 2014, to appear.

[75] M. G. McGaffin, S. Ramani, and J. A. Fessler, "Reduced memory augmented Lagrangian algorithm for 3D iterative X-ray CT image reconstruction," in *Proc. SPIE 8313 Medical Imaging 2012: Phys. Med. Im.*, 2012, p. 831327.

[76] A. Mehranian, A. Rahmim, M. R. Ay, F. Kotasidis, and H. Zaidi, "An ordered-subsets proximal preconditioned gradient algorithm for edge-preserving PET image reconstruction," *Med. Phys.*, vol. 40, no. 5, p. 052503, 2013.

[77] D. Modgil, A. M. Alessio, M. D. Bindschadler, K. J. Little, D. Rigie, P. A. Vargas, and P. J. La Rivière, "Multi-dimensional sinogram restoration for myocardial blood flow estimation from dose-reduced dynamic CT," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 217–20.

[78] A. Nemirovski and D. Yudin, *Problem complexity and method efficiency in optimization*. John Wiley, 1983.

[79] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Dokl. Akad. Nauk. USSR*, vol. 269, no. 3, pp. 543–7, 1983.

[80] ——, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Math. Dokl.*, vol. 27, no. 2, pp. 372–76, 1983.

[81] ——, *Introductory lectures on convex optimization: A basic course*. Kluwer, 1984.

[82] ——, "On an approach to the construction of optimal methods of minimization of smooth convex functions," *Èkonom. i. Mat. Metody*, vol. 24, pp. 509–17, 1988.

[83] ——, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–52, May 2005.

[84] ——, "Gradient methods for minimizing composite objective function," CORE, Catholic University of Louvain, Louvain-la-Neuve, Belgium, Tech. Rep., 2007.

[85] H. Nien and J. A. Fessler, "Combining augmented Lagrangian method with ordered subsets for X-ray CT image reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 280–3.

[86] ——, "Fast splitting-based ordered-subsets X-ray CT image reconstruction," in *Proc. 3rd Intl. Mtg. on image formation in X-ray CT*, 2014, to appear.

[87] ——, "Fast X-ray CT image reconstruction using the linearized augmented Lagrangian method with ordered subsets," *IEEE Trans. Med. Imag.*, 2014, submitted.

[88] F. Noo, M. Defrise, and R. Clackdoyle, "Single-slice rebinning method for helical cone-beam CT," *Phys. Med. Biol.*, vol. 44, no. 2, pp. 561–70, Feb. 1999.

[89] J. Nuyts, B. De Man, P. Dupont, M. Defrise, P. Suetens, and L. Mortelmans, "Iterative reconstruction for helical CT: A simulation study," *Phys. Med. Biol.*, vol. 43, no. 4, pp. 729–37, Apr. 1998.

[90] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Found. Computational Math.*, 2014.

[91] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. New York: Academic, 1970.

[92] P. J. Pickhardt, M. G. Lubner, D. H. Kim, J. Tang, J. A. Ruma, A. M. del Rio, and G.-H. Chen, "Abdominal CT with model-based iterative reconstruction (MBIR): Initial results of a prospective Trial comparing ultralow-dose with standard-dose imaging," *Am. J. Roentgenol.*, vol. 199, Dec. 2012.

[93] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comp. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[94] S. Ramani and J. A. Fessler, "A splitting-based iterative algorithm for accelerated statistical X-ray CT reconstruction," *IEEE Trans. Med. Imag.*, vol. 31, no. 3, pp. 677–88, Mar. 2012.

[95] J. M. Rosen, J. Wu, T. F. Wenisch, and J. A. Fessler, "Iterative helical CT reconstruction in the cloud for ten dollars in five minutes," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2013, pp. 241–4.

[96] K. Sauer and C. Bouman, "A local update strategy for iterative reconstruction from projections," *IEEE Trans. Sig. Proc.*, vol. 41, no. 2, pp. 534–48, Feb. 1993.

[97] K. D. Sauer, S. Borman, and C. A. Bouman, "Parallel computation of sequential pixel updates in statistical tomographic reconstruction," in *Proc. IEEE Intl. Conf. on Image Processing*, vol. 3, 1995, pp. 93–6.

[98] W. P. Segars, M. Mahesh, T. J. Beck, E. C. Frey, and B. M. W. Tsui, "Realistic CT simulation using the 4D XCAT phantom," *Med. Phys.*, vol. 35, no. 8, pp. 3800–8, Aug. 2008.

[99] S. Sotthivirat and J. A. Fessler, "Image recovery using partitioned-separable paraboloidal surrogate coordinate ascent algorithms," *IEEE Trans. Im. Proc.*, vol. 11, no. 3, pp. 306–17, Mar. 2002.

[100] S. Srivastava, "Accelerated statistical image reconstruction algorithms and simplified cost functions for X-ray computed tomography," Ph.D. dissertation, Univ. of Michigan, Ann Arbor, MI, 48109-2122, Ann Arbor, MI, 2008.

[101] J. W. Stayman, Y. Otake, J. L. Prince, A. J. Khanna, and J. H. Siewerdsen, "Model-based tomographic reconstruction of objects containing known components," *IEEE Trans. Med. Imag.*, vol. 31, no. 10, pp. 1837–48, Oct. 2012.

[102] J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "A recursive filter for noise reduction in statistical iterative tomographic imaging," in *Proc. SPIE 6065 Computational Imaging IV*, 2006, p. 60650X.

[103] 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.*, vol. 34, no. 11, pp. 4526–44, Nov. 2007.

[104] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," 2008.

[105] ——, "Approximation accuracy, gradient methods, and error bound for structured convex optimization," *Mathematical Programming*, vol. 125, no. 2, pp. 263–95, 2010.

[106] J. Weickert, B. M. H. Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Trans. Im. Proc.*, vol. 7, no. 3, pp. 398–410, Mar. 1998.

[107] M. Yavuz and J. A. Fessler, "Statistical image reconstruction methods for randoms-precorrected PET scans," *Med. Im. Anal.*, vol. 2, no. 4, pp. 369–78, Dec. 1998.

[108] L. Yu, Y. Zou, E. Y. Sidky, C. A. Pelizzari, P. Munro, and X. Pan, "Region of interest reconstruction from truncated data in circular cone-beam CT," *IEEE Trans. Med. Imag.*, vol. 25, no. 7, pp. 869–81, Jul. 2006.

[109] Z. Yu, J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "Fast model-based X-ray CT reconstruction using spatially non-homogeneous ICD optimization," *IEEE Trans. Im. Proc.*, vol. 20, no. 1, pp. 161–75, Jan. 2011.