Iterative Reconstruction Algorithms for Polyenergetic X-Ray Computerized Tomography

by

Nargol Rezvani

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Iterative Reconstruction Algorithms for Polyenergetic X-Ray Computerized Tomography

Nargol Rezvani

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2012

A reconstruction algorithm in computerized tomography is a procedure for reconstructing the attenuation coefficient, a real-valued function associated with the object of interest, from the measured projection data. Generally speaking, reconstruction algorithms in CT fall into two categories: direct, e.g., filtered back-projection (FBP), or iterative. In this thesis, we discuss a new fast matrix-free iterative reconstruction method based on a polyenergetic model.

While most modern x-ray CT scanners rely on the well-known filtered back-projection algorithm, the corresponding reconstructions can be corrupted by beam hardening artifacts. These artifacts arise from the unrealistic physical assumption of monoenergetic x-ray beams. In this thesis, to compensate, we use an alternative model that accounts for differential absorption of polyenergetic x-ray photons and discretize it directly. We do not assume any prior knowledge about the physical properties of the scanned object. We study and implement different solvers and nonlinear unconstrained optimization methods such as a Newton-like method and an extension of the Levenberg-Marquardt-Fletcher algorithm. We explain how we can use the structure of the Radon matrix and the properties of FBP to make our method matrix-free and fast. Finally, we discuss how we regularize our problem by applying different regularization methods, such as Tikhonov and regularization in the 1-norm. We present numerical reconstructions based on the associated nonlinear discrete formulation incorporating various iterative optimization methods.

Dedication

To my parents, Taraneh and Mahmoud.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my PhD supervisors Dr. Ken Jackson and Dr. Dhavide Aruliah for their continuous support, patience, motivation, enthusiasm, and immense knowledge. Indubitably, this thesis would have not been possible without them.

I also owe my most profound gratitude to Dr. Rob Corless, my MSc supervisor, who has been a great inspiration to me. He challenged me beyond my expectations and made me discover who I really am and where I stand.

It has been an honor to be their student. They all portrayed what the well-known Persian poet, Sa'adi, said more than seven hundred years ago:

"All men are members of the same body
Created from one essence
Should destiny bring agony to one
Others, restless, will suffer disturbance
Thou who remainest indifferent to others' miseries
Meritest not the human competence"[1]

Besides my supervisors, I would like to thank my external examiner, Dr. Adel Faridani. His insightful comments and helpful suggestions brought a new spirit to this thesis. I would also like to thank the rest of my thesis committee: Dr. Christina Christara, Dr. Wayne Enright and Dr. Tom Fairgrieve for their encouragement, perceptive comments, and hard questions.

My sincere thanks also go to Dr. Douglas Moseley, Dr. Jeff Siewerdsen, and Dr. David Jaffray for offering me a Summer internship at the Princess Margaret Hospital. Their help and guidance is very much appreciated.

[1]Translated from Farsi by Abdolmahmoud Rezvani

I am truly thankful to my undergraduate professors, Dr. Hassan Haghighi, Dr. Fereshteh Malek and Dr. Kourosh Norouzi, who changed my perspective toward mathematics.

I owe my sincere thankfulness to my high school teachers, Ms. Mahrokh Khoshnevis and Ms. Elham Nejati, who taught me how to be a stronger woman and have faith in myself.

My most genuine gratefulness and respect go to Ms. Afzali, my first grade teacher, who taught me that in the process of learning, nothing can take the place of persistence.

Furthermore, I would like to thank the Department of Computer Science at the University of Toronto, the Department of Applied Mathematics at the University of Western Ontario, and the Department of Mathematics at the K. N. Toosi University of Technology. I would also like to thank MITACS (the Mathematics of Information Technology and Complex Systems) Canadian research network and AAPM (the American Association of Physicists in Medicine) for sponsoring my internship project. In addition, I thank the Walter C. Sumner foundation and NSERC (the Natural Sciences and Engineering Research Council of Canada) for partly sponsoring my PhD project.

I wish to thank my dear family and friends for their love and support. In particular, I would like to thank (alphabetically ordered) Afsar Shirazi, Dr. Azar Shakoori, Behdad Esfahbod, Farbod Nassiri, Dr. Houman Javidnia, Dr. Jingrui Zhang, Kataneh Karimian, Mahsa Moallem, Meg Kwasnicki, Narges Nattaghi, Shohreh Armin, Dr. Solmaz Kolahi, Sumaya Corless, Taban Karimian, Termeh Karimian and many more.

Lastly, I would like to thank my close family. My utmost gratitude goes to my mother, Taraneh Karimian, who has always been my greatest inspiration in life. She has showed me the true meaning of unconditional love. I will always be grateful to my father, Dr. Abdolmahmoud Rezvani, for his strong faith in me and his tremendous support at every stage of my life. He proved to me, by his actions, that following one's passion, even if one fails, brings happiness; but chasing wealth may not. I am thankful for the presence of my little, yet bighearted, sister, Sogol, in my life. I would have not been

the person I am without her. I would also like to express my most sincere appreciation to my grandparents, my ultimate sources of pure love and kindness, Molouk Ghaffari (Aziz), Parvindokht Baghdadi (Papari), Masoumeh Mahmoudi and Dr. Iradj Karimian. Finally, I am very grateful for having my loving and patient husband, Rubil, in my life. I appreciate his understanding and faithful support during the final stages of this PhD very much.

List of Symbols

Symbol	Description
μ	Continuous attenuation coefficient
$\boldsymbol{\mu}$	Discretized attenuation coefficient
$\mathcal{R}$ and $\mathbf{R}$	Radon transform and Radon matrix
$\mathcal{R}^*$ and $\mathbf{R}^*$	Adjoint of Radon transform and Adjoint of Radon matrix
$\widehat{\mu}$	Fourier transform of μ
$\widetilde{h}(r,\theta)$	1-D Fourier transform, in the affine parameter, of function $h(t,\theta)$
$\mathbf{F}$	Fourier transform matrix
N_t	Number of affine parameter values
N_θ	Number of projection angles
N_{grid}	Number of grid points (pixels or voxels)
N_ε	Number of energy levels
N_{proj}	Number of projections, which is $N_t \times N_\theta$
$P_{t,\theta}$	Projection data (sinogram)
P_0	Blank scans (intensity without the presence of an object)
ε	Beam energy
$\langle \mathbf{x}, \mathbf{y} \rangle$	Inner product of vectors $\mathbf{x}$ and $\mathbf{y}$
$\widehat{g}$	Filter in the frequency domain
$\mathbf{D}^{\widehat{\mathbf{g}}}$	Filtering matrix based on the filter $\widehat{g}$
$\mathbf{J}$	Interpolation matrix
w	Quadrature weights

Contents

List of Figures

Chapter 1

Introduction

X-ray Computerized Tomography[1] (CT) is a nondestructive technique for representing interior features within solid objects from a series of x-ray measurements taken from different angles around the object. CT has become an important tool in medical imaging. It has had a considerable impact on the diagnosis of many different diseases. Besides medicine, it also has industrial applications, such as nondestructive materials testing.

In tomographic imaging, an object is illuminated with a series of x-rays from multiple orientations. Then the change in the intensity of x-rays along a series of linear paths is measured. This change is characterized by Beer's Law (or alternative physical models – see Sections 2.1 and 3.3.1), which describes intensity change as a function of x-ray energy, path length, and material *attenuation coefficient*, a real-valued function defined on $\mathbb{R}^2$ or $\mathbb{R}^3$ that measures how much the object can absorb or scatter x-rays of a given energy. Then a specialized algorithm is used to reconstruct the image in question.

The elements of data acquisition in x-ray tomography are an x-ray source, a series of detectors that measure x-ray intensity attenuation along multiple beams, and a rotational geometry with respect to the object being imaged (see Figure 1.1). Different configurations of these components can be used to create CT scanners (see Section 2.3).

[1]Literally speaking, tomography means *slice imaging*. The word *tomography* is derived from the Greek words tomos meaning slice, and graphein meaning to write.

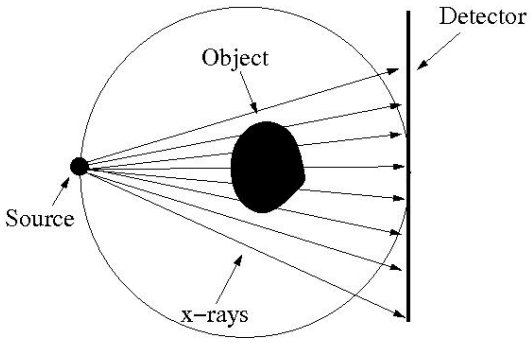


Figure 1.1: Physical model

The first CT scanner was constructed by Godfrey Newbold Hounsfield in 1972. Independently and at about the same time, Allan McLeod Cormack designed a similar process. Hounsfield and Cormack shared the 1979 Nobel Prize in Medicine. The first CT x-ray machine was installed in Atkinson Morley's Hospital in Wimbledon, England, and the first patient brain-scan was made with it in 1972.

A CT reconstruction algorithm is a procedure for reconstructing the attenuation coefficient from the measured projection data. Generally speaking reconstruction algorithms in CT fall into two categories: direct, e.g., filtered back-projection (FBP), or iterative, e.g., algebraic reconstruction techniques (ART) or statistical image reconstruction techniques (SIRT).

Filtered back-projection is the most commonly used reconstruction algorithm in CT. This method is based on the Central slice theorem. In Chapter 2, we cover the mathematical background of x-ray CT, introduce some basic notation and definitions and present the formulation of FBP.

Using FBP, we develop a software package called `OSCaR` (Open Source Cone-beam Reconstructor) for generating 3-D reconstructions from x-ray data acquired from cone-beam scanning geometries, in MATLAB. `OSCaR` is based on the well-known Feldkamp-Davis-Kress (FDK) reconstruction algorithm for 3-D cone-beam CT [21]. In Appendix A, we describe `OSCaR`.

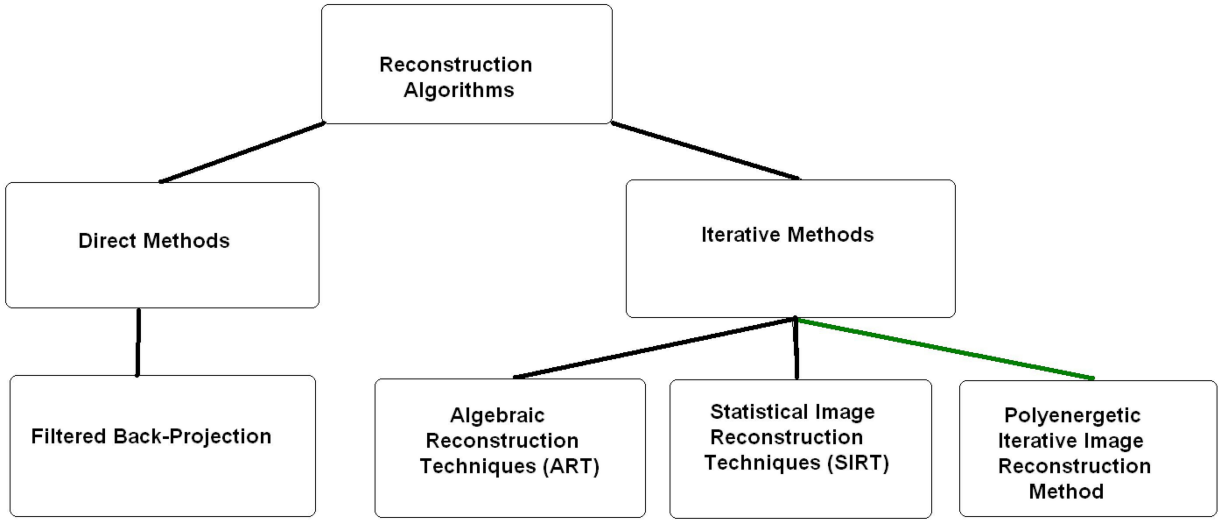


Figure 1.2: Diagram of reconstruction algorithms in CT.

Recently, the limits of FBP have been challenged by the progress toward scanning geometries, such as cone-beam and multi-slice helical CT. Therefore, there has been more interest in iterative methods for CT image reconstruction such as ART and SIRT. Iterative methods can have several advantages over direct methods. They can incorporate prior knowledge, including system geometry, detector response and object constraints. Iterative methods also permit modeling data noise. An assumption underlying FBP is that x-ray sources are monoenergetic; in practice, there is a nonuniform distribution of photons of different wavelengths that leads to a phenomenon physicists call beam hardening [19, p. 68]. Some iterative methods, such as statistical methods can model polyenergetic x-ray sources and thus account for beam hardening in the reconstruction [18, 37]. However, iterative techniques are widely perceived to require more computing time than

FFT-based FBP techniques [6, 18, 19, 28, 37]. We discuss different iterative methods in Chapter 3.

In Chapter 4, we present a novel non-statistical iterative reconstruction method that is based on a polyenergetic model. We apply quadrature rules, e.g., a Gaussian quadrature rule or the composite Trapezoidal rule, and take advantage of some properties of the Radon transform, FBP and ART methods to discretize the energy-dependent continuous model with respect to different energy levels and different projections. In this model, we must account for the nonlinearity in the measurements which is caused by the energy-dependent attenuation coefficient. To this end, we have studied different solvers and nonlinear unconstrained optimization methods such as a Newton-like method and an extension of the Levenberg-Marquardt-Fletcher algorithm. This polyenergetic reconstruction method is one of the major contributions of this thesis.

In the first variant of our iterative reconstruction method, we need to store large sparse, but not very sparse, matrices. Therefore, the method may either experience long computation time, or even worse, run out of memory. However, if we use small data sets, we get results with better image quality than the corresponding FBP solutions. Unfortunately, clinical data sets are usually very large, and therefore, a naive implementation of our reconstruction method will not be effective. This motivates the development of a matrix-free fast algorithm for large scale problems.

In Chapter 5, we analyze the structure of the Radon matrix to exploit the structure of the Jacobian matrix associated with our objective function. Then we apply the properties of the Radon transform and FBP to make our computations matrix-free and fast. Moreover, this approach allows us to take advantage of existing, highly efficient implementations for FBP and the Radon transform. This development of a fast matrix-free implementation for the polyenergetic model is another major contribution of this thesis.

1.1 Main contributions

The main contributions of this thesis (primarily described in Chapters 4 and 5) are as follows.

- We introduce and discuss a new iterative polyenergetic reconstruction method that is based on a polyenergetic model which accounts for beam hardening. To develop this method, we use an extension of the Levenberg-Marquardt-Fletcher algorithm.
- Our polyenergetic model has two integrals. We apply quadrature rules, e.g., a Gaussian quadrature or the composite Trapezoidal rule, to discretize the outer integral with respect to different energy levels. At this step we use a typical x-ray spectrum (see Figure 4.1). To discretize the inner integral with respect to different projection angles and affine parameters, we take advantage of some properties of the Radon transform and FBP. At this step we may also use the ART instead.
- We use the known structure of the Radon matrix to exploit the structure of the Jacobian matrix of our objective function (see Figure 5.1). This helps us gain insight into our problem.
- We take advantage of well-known and fast implementations of the Radon transform and the FBP method to make our iterative polyenergetic method matrix-free and fast. The matrix representation of FBP introduced in Section 2.5 helps us understand this process better.
- We introduce several stabilizing operators and regularize the problem using Tikhonov regularization and regularizing in the 1-norm.
- We discuss the development of a software package called `OSCaR` (Open Source Cone-beam Reconstructor) for generating 3-D reconstructions from x-ray data acquired from cone-beam scanning geometries, in Appendix A. This software is written in Matlab and uses the well-known Feldkamp-Davis-Kress (FDK) algorithm [21].

Chapter 2

Mathematical Background and Filtered Back-Projection

2.1 A physical model

X-ray slice data is generated using an x-ray source that rotates around the object; x-ray sensors are positioned on the opposite side of the object from the x-ray source (see Figure 1.1).

Objects of interest in x-ray imaging are associated with a real-valued function defined on $\mathbb{R}^2$ (or $\mathbb{R}^3$), called the *attenuation coefficient*. This value measures how much the object can absorb or scatter x-ray of a given energy. This function varies from point to point within the object and is usually taken to vanish outside it. For example, bone has a much higher attenuation coefficient than soft tissue.

Sometimes the attenuation coefficient is compared to the attenuation coefficient of water and quoted in terms of a dimensionless quantity called the *Hounsfield* unit (HU). The normalized attenuation coefficient, in Hounsfield units, is defined by

$$H_{\text{tissue}} = \frac{\mu_{\text{tissue}} - \mu_{\text{water}}}{\mu_{\text{water}}} \times 1000. \tag{2.1}$$

In a typical clinical situation the attenuation coefficients range between -1000 HU (air) and 1100 HU (bone). [19]

Here we describe a very common model for the interaction of x-rays with matter in CT. For this model, there are three basic assumptions [19]:

- Straight lines (no refraction/diffraction): x-ray beams travel along straight lines and are not bent by the objects they pass through.

- Monoenergetic source: The waves constituting the x-ray beam are all of the same energy (or, equivalently, the same wavelength).

- Beer's Law: Each material considered has a characteristic linear attenuation coefficient μ for x-rays of a given energy. The intensity, P, of the x-ray beam satisfies Beer's law

$$\frac{dP}{ds} = -\mu P \quad \Rightarrow \quad \int_l \mu \, ds = -\ln\left(\frac{P}{P_0}\right). \tag{2.2}$$

  Here s is the arc-length along the straight-line trajectory l of the x-ray beam and P_0 is the blank scan intensity.

2.2 Mathematical background of x-ray CT

In this section we briefly describe the mathematical background for x-ray CT, based mostly on references [19] and [34]. We also explain how to estimate the attenuation coefficient, which is the key component of image reconstruction.

2.2.1 The Radon transform

In medical imaging, the *Radon transform* of a function provides a mathematical model for the measured attenuation. The Radon transform of function μ is the totality of line integrals of μ along the lines in the plane. More specifically the Radon transform of

$\mu : \mathbb{R}^2 \longrightarrow \mathbb{R}$ (in 3-D images, $\mu : \mathbb{R}^3 \longrightarrow \mathbb{R}$) is defined by

$$[\mathcal{R}\mu(\mathbf{x})](t,\theta) := \int_{l_{t,\theta}} \mu(\mathbf{x})\, ds, \tag{2.3}$$

where

$$l_{(t,\theta)} = \{\mathbf{x} : \langle \mathbf{x}, \omega(\theta)\rangle = t\} = \{t\boldsymbol{\omega}(\theta) + s\widehat{\boldsymbol{\omega}}(\theta) \ : \ s \in \mathbb{R}\}\,. \tag{2.4}$$

Here $\widehat{\boldsymbol{\omega}}(\theta) = (-\sin\theta, \cos\theta)$ is the unit vector perpendicular to the unit vector $\boldsymbol{\omega}(\theta) = (\cos\theta, \sin\theta)$. This parametric representation of the line gives the following formula.

$$\mathcal{R}\mu(t,\theta) := \int_{\mathbb{R}} \mu(t\boldsymbol{\omega}(\theta) + s\widehat{\boldsymbol{\omega}}(\theta))\, ds. \tag{2.5}$$

In terms of Cartesian coordinates in $\mathbb{R}^2$ and (t,θ)-coordinates for the set of oriented lines, (2.5) can be re-expressed as

$$\mathcal{R}\mu(t,\theta) := \int_{\mathbb{R}} \mu(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\, ds. \tag{2.6}$$

We say the function μ is in the *natural domain* of the Radon transform if it satisfies

$$|\mathcal{R}\mu(t,\theta)| := \left| \int_{\mathbb{R}} \mu(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\, ds \right| < \infty \qquad \text{for all } (t,\theta) \in \mathbb{R}\times[0,2\pi]\,. \tag{2.7}$$

This is really two different conditions:

1. The function is sufficiently regular that restricting it to any line gives a locally integrable function.

2. The function goes to zero rapidly enough for the improper integrals to converge.

The variable t is called the *affine* parameter; it is the oriented distance from the line $l_{(t,\theta)}$ to the origin.

2.2.2 Fourier transform

The *Fourier transform* of an L^1-function[1] μ, defined on $\mathbb{R}$, is the function $\widehat{\mu}$ defined on $\mathbb{R}$ by the integral

$$\widehat{\mu}(t) = \int_{-\infty}^{+\infty} \mu(x) e^{-ixt}\, dx. \tag{2.9}$$

The utility of the Fourier transform comes from the fact that μ can be reconstructed from $\widehat{\mu}$ using the *Fourier Inversion Formula* defined in (2.10) below. Suppose that μ is an L^1-function such that $\widehat{\mu}$ is also in $L^1(\mathbb{R})$. Then

$$\mu(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{\mu}(t) e^{ixt}\, dt. \tag{2.10}$$

The above formulae can be extended to $\mathbb{R}^n$. Suppose that μ is an L^1-function defined on $\mathbb{R}^n$. The Fourier transform $\widehat{\mu}$ of μ is defined by

$$\widehat{\mu}(\mathbf{t}) = \int_{\mathbb{R}^n} \mu(\mathbf{x}) e^{-i\langle \mathbf{t}, \mathbf{x} \rangle}\, d\mathbf{x}, \tag{2.11}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathbb{R}^n$. If $\widehat{\mu}$ also belongs to $L^1(\mathbb{R}^n)$, then

$$\mu(\mathbf{x}) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} \widehat{\mu}(\mathbf{t}) e^{i\langle \mathbf{x}, \mathbf{t} \rangle}\, d\mathbf{t}. \tag{2.12}$$

[1] Let μ be a function defined on $\mathbb{R}^n$. We say that μ is *absolutely integrable* or an L^1-function if

$$\|\mu\|_1 = \int_{\mathbb{R}^n} |\mu(\mathbf{x})|\, dx < \infty \tag{2.8}$$

2.2.3 Central slice theorem

The *Central slice theorem*, also known as the *Fourier slice theorem*, states an important relation between the Fourier transform and the Radon transform. This Theorem is key to efficient tomographic imaging, relating the measured projection data to the two-dimensional Fourier transform of the object cross section.

Theorem 2.2.1. ***Central slice theorem.*** *Let μ be an absolutely integrable function in the natural domain of the Radon transform. For any real number r and unit vector $\boldsymbol{\omega}$, we have the identity*

$$\int_{-\infty}^{+\infty} [\mathcal{R}\mu(\mathbf{x})](t,\theta)e^{-itr}\,dt = \widehat{\mu}(r\boldsymbol{\omega}(\theta)). \tag{2.13}$$

The following example shows that, according to the Central slice theorem, the 2-D Fourier transform, $\widehat{\mu}(r\boldsymbol{\omega}(\boldsymbol{\theta}))$, is the 1-D Fourier transform of $\mathcal{R}\mu(t,\theta)$ with respect to t.

Example 2.2.1. *Let $\boldsymbol{\omega}(\theta) = (1,0)$ and $\widehat{\boldsymbol{\omega}}(\theta) = (0,1)$. The Radon transform at (t,θ) is given by*

$$\begin{aligned}\mathcal{R}\mu(t,\theta) &= \int_{-\infty}^{+\infty} \mu(t\boldsymbol{\omega}(\theta) + s\widehat{\boldsymbol{\omega}}(\theta))\,ds \\ &= \int_{-\infty}^{+\infty} \mu(t,s)\,ds.\end{aligned}$$

The 1-D Fourier transform of $\mathcal{R}\mu(t,\theta)$ with respect to t is

$$\int_{-\infty}^{+\infty} \mathcal{R}\mu(t,\theta)e^{-irt}\,dt = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mu(t,s)e^{-irt}\,ds\,dt.$$

On the other hand, the 2-D Fourier transform of μ is

$$\begin{aligned}\widehat{\mu}(r\boldsymbol{\omega}(\theta)) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mu(t,s)e^{-i\langle r\boldsymbol{\omega},(t,s)\rangle}\,ds\,dt \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mu(t,s)e^{-irt}\,ds\,dt.\end{aligned}$$

2.3 Acquisition geometries

Different geometries for acquisition of data are used in modern CT scanners. That is, sampling the Radon transform of the attenuation coefficient and, consequently, the algorithmic implementation of the filtered back-projection formula varies in different machines with different designs, e.g., parallel-beam, fan-beam or cone-beam.

2.3.1 Parallel-beam

In a parallel-beam scanner, a source-detector combination must linearly scan over the length of the projection, then rotate a certain angular interval, then scan linearly over the length of the next projection, and so on. In these machines, the approximate samples of $\mathcal{R}\mu$ are measured in a finite set of directions. The measurements made in a given direction are therefore samples of $\mathcal{R}\mu$ at a set of equally spaced affine parameters. Figure 2.1 shows the geometry of the parallel projection.

2.3.2 Fan-beam

A fan-beam machine has a point source on one side of the object and a bank of detectors, which collects the measurements in one fan simultaneously, on the other side. The source and the entire bank of detectors are rotated to generate the desired number of fan projections. The source is pulsed at a discrete sequence of angles, and the measurements of $\mathcal{R}\mu$ are collected for a finite family of lines passing through the source. There are two

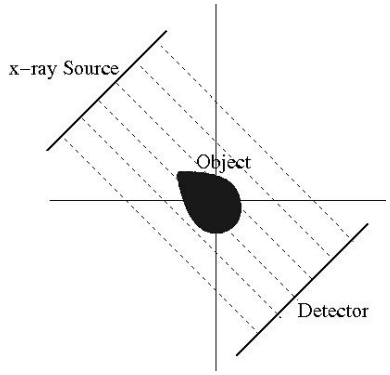


Figure 2.1: Parallel projection.

different designs for fan-beam machines:

- **Equiangular fan-beam CT** In such machines, projections are sampled at equiangular intervals, i.e., the angle between rays is constant but the detector spacing is uneven. In this case the detectors for the measurement of the line integrals are arranged on a straight line (see Figure 2.2).

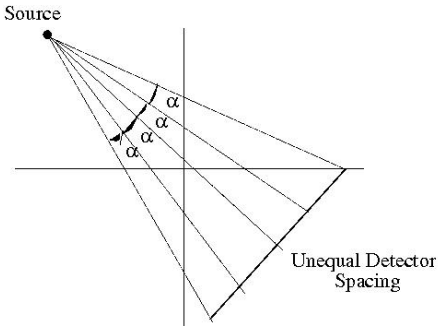


Figure 2.2: Equiangular fan-beam projection.

- **Equispaced fan-beam CT** In such machines, the measurements are made at equispaced intervals. In this case the detectors are arranged with constant spacing along a line but the angle between rays is not constant (see Figure 2.3).

Of course these two are not mutually exclusive. If the detectors are arranged on the arc of a circle whose centre is at the source, rather than a straight line, then it is possible to have both an equispaced and an equiangular detector (see Figure 2.4).

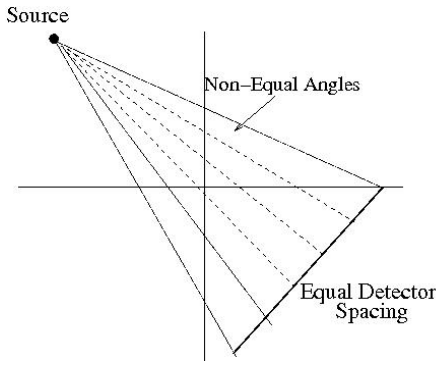


Figure 2.3: Equispaced fan-beam projection.

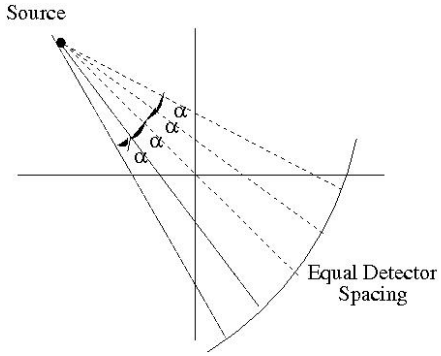


Figure 2.4: Equiangular and equispaced fan-beam projection.

2.3.3 Cone-beam

An efficient way to image a three-dimensional object is a generalization of the two-dimensional fan-beam algorithms. Instead of illuminating a slice of the object with a fan of x-rays, the entire object is illuminated with a point source and the x-ray flux is measured using a planar detector (see Figure 2.5). This is usually called cone-beam reconstruction as the x-rays are assumed to be emitted from the source in a three-dimensional cone-shaped beam. The object is positioned between the x-ray source and the planar detector. There are two distinct ways to gather projection data from different angles during a scan:

1. The object is rotated about an axis and the source-detector pair is held fixed in space; and

2. The object is held fixed in space while the source-detector pair is rotated about the axis.

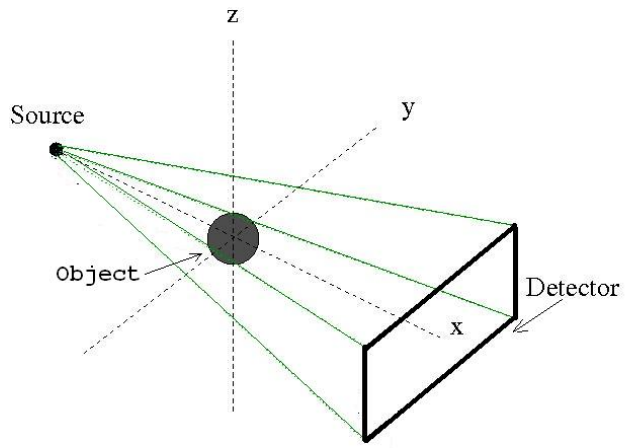


Figure 2.5: Cone-beam projection.

The second option is most commonly used in clinical C-arm CT scanners. Moreover, although the axis in Figure 2.5 is vertical, most clinical scanners use a horizontal axis of rotation (for patients lying down). The principal advantages of a cone-beam x-ray scanner are in reducing data collection time during the scan as well as maximizing the use of x-ray photons actually passing through the object, thereby reducing patient dose [34].

2.4 Direct reconstruction methods: Filtered back-projection

The Central slice theorem and the inversion formula for the Fourier transform give an inversion formula for the Radon transform, often called the *filtered back-projection* (FBP) formula. FBP can be implemented with fast Fourier transforms.

To explain how this can be done, we first introduce notation for the 1-D Fourier transform, in the affine parameter, of a function $h(t, \theta)$:

$$\widetilde{h}(r, \theta) = \int_{-\infty}^{+\infty} h(t, \theta) e^{-itr}\, dt. \tag{2.14}$$

Applying this to the Radon transform, we get from Theorem 2.2.1

$$\widetilde{\mathcal{R}\mu}(r, \theta) = \int_{-\infty}^{+\infty} \mathcal{R}\mu(t, \theta) e^{-itr}\, dt = \widehat{\mu}(r\boldsymbol{\omega}(\theta)). \tag{2.15}$$

Theorem 2.4.1. *(Radon inversion formula): If $\mu = \mu(\mathbf{x})$ is an absolutely integrable function in the natural domain of the Radon transform and $\widehat{\mu}$ is also absolutely integrable, then*

$$\mu(\mathbf{x}) = \frac{1}{[2\pi]^2} \int_0^{\pi} \int_{-\infty}^{+\infty} e^{ir\langle \mathbf{x}, \boldsymbol{\omega}(\theta)\rangle} \widetilde{\mathcal{R}\mu}(r, \theta) |r|\, dr\, d\theta. \tag{2.16}$$

The Radon inversion formula (2.16) describes the computation of a function μ from its Radon transform $\mathcal{R}\mu$. It is a starting point for practical algorithms. The most obvious flaw in this model is that, in practice, $\mathcal{R}\mu(t, \theta)$ can only be measured for a finite set of pairs (t, θ).

The filtered back-projection formula can be decomposed into two stages (corresponding to the two nested integrals in (2.16)).

- The first stage is the application of a linear, shift-invariant[2] filter to the Radon transform (which corresponds to integration with respect to the variable r in (2.16)).

[2] A filter g is shift invariant (mapping functions defined on $\mathbb{R}^n$ to functions defined on $\mathbb{R}^n$) if it satisfies

$$g(f_t) = (gf)_t, \tag{2.17}$$

where for $t \in \mathbb{R}^n$ the shift of f by t is the function f_t, defined by

$$f_t(x) = f(x - t). \tag{2.18}$$

- The second stage is an angular back-projection (BP) of the filtered Radon transform (which corresponds to integration with respect to the variable θ in (2.16)). In practice, this step is usually computed using an interpolation, e.g. nearest-neighbor interpolation, linear interpolation, spline interpolation, or shape-preserving piecewise cubic interpolation.

Note that the projection data, which is also known as the *sinogram data*, gives us $\mathcal{R}\mu(t,\theta)$. The projection data is often represented in (t,θ)-space. The data is often called a *sinogram* since each object point contributes to a projection value along a sinusoidal curve in this space (see Figure 2.6). We first compute $\widetilde{\mathcal{R}\mu}(r,\theta)$ by (2.15) and then we compute the double integral in (2.16) in two steps, as described above.

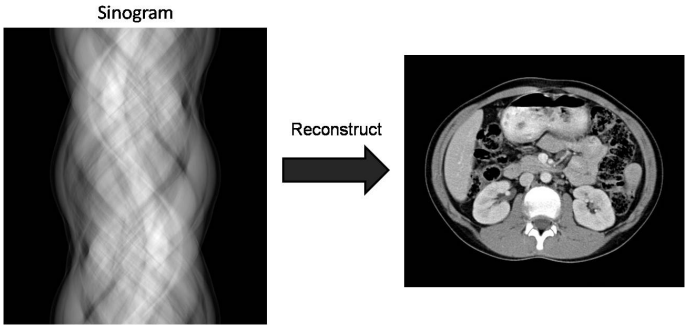


Figure 2.6: Reconstruction algorithms convert raw data (sinogram) to an image of the object. This image is taken from `http://scien.stanford.edu/class/psych221/projects/08/AdamWang/project_report.htm`

2.4.1 More about filtering

In electrical engineering and signal/image processing, filters are used to transform signals from one form to another, specifically to eliminate (filter out) various frequencies in a signal [27]. Mathematically speaking, a filter is a map from one space of functions to another. Filtering is usually needed to compensate for noise in data. A common approach

for extracting the desired signals from the raw data is to convolve[3] the filter with the Radon transform of the input data. In practice, functions representing the filters are usually defined in the frequency (Fourier) domain. Thus, the filtering is performed as a multiplication in the frequency domain instead of convolution in the spatial domain. For more details, refer to [53].

Many common filters used in FBP consist of a ramp (or Ram-Lak) filer multiplied by an apodizing window, $\mathrm{WIN}(r)$, i.e.,

$$\widehat{g}(r) = \mathrm{WIN}(r)\ |r|. \tag{2.19}$$

The purpose of the apodizing window is to localize the support of the filter in the frequency domain (thereby suppressing noise). Common apodizing windows include:

- Hamming: $\mathrm{WIN}(r) = (0.54 + 0.46\cos(\frac{\pi r}{B_{\mathrm{win}}}))\chi_{[-B_{\mathrm{win}},B_{\mathrm{win}}]}(r)$,
- Hann: $\mathrm{WIN}(r) = (0.5 + 0.5\cos(\frac{\pi r}{B_{\mathrm{win}}}))\chi_{[-B_{\mathrm{win}},B_{\mathrm{win}}]}(r)$,

where the *characteristic function*, $\chi_{[-B_{\mathrm{win}},B_{\mathrm{win}}]}$, is

$$\chi_{[-B_{\mathrm{win}},B_{\mathrm{win}}]}(r) = \begin{cases} 1 & \text{if } r \in [-B_{\mathrm{win}}, B_{\mathrm{win}}], \\ 0 & \text{otherwise.} \end{cases} \tag{2.20}$$

For more details about windows and filters, refer to [19, 43, 58]. Also, see [60] for an interesting discussion on the definition, properties, and construction of an idealised ramp filter.

[3] If f is an L^1-function defined on $\mathbb{R}^n$ and g is a bounded, locally integrable function, the *convolution product* of f and g is the function on $\mathbb{R}^n$ defined by the integral

$$f * g(x) = \int_{\mathbb{R}^n} f(x-y)g(y)\,dy.$$

2.4.2 More about back-projection

In the back-projection stage, the projection value is back-projected, or smeared out over the image points along a particular ray. That is, for each pixel or voxel in the reconstruction domain, after applying the desired filter in the Radon transform, we compute the angular integral of the Radon inversion formula.

In the exact formula (2.3) for the Radon transform, it is assumed that $\mathcal{R}\mu$ is known for all lines; in practice, $\mathcal{R}\mu$ is approximately sampled for a finite number of lines. The Radon inversion formula (2.16) leads to methods for reconstructing discrete approximations of the attenuation coefficient from a realistic finite collection of measurements. The computation step in FBP depends on the acquisition method, such as parallel-beam, fan-beam, or cone-beam, discussed in Section 2.3. Here we describe an algorithm for parallel-beam geometry, which is inspired by similar discussions in [19, 34].

2.4.3 A Reconstruction algorithm for parallel-beam machines

In parallel-beam scanners, the samples of $\mathcal{R}\mu$ are measured at a finite set of equally spaced angles and equally spaced affine parameters (see Figure 2.7).

- **Projection angle.** The samples are measured at a finite set of directions associated with equally spaced angles,

$$\{\boldsymbol{\omega}(k\Delta\theta), k = 0, \ldots, M\}, \tag{2.21}$$

  where

$$\Delta\theta = \frac{\pi}{M+1}, \tag{2.22}$$

  and, as before,

$$\boldsymbol{\omega}(k\Delta\theta) = (\cos(k\Delta\theta), \sin(k\Delta\theta)). \tag{2.23}$$

- **Affine Parameter.** Then for a given direction we sample at a finite set of equally

spaced affine parameters $t \in [-L, L]$. That is,

$$\{t = jd : j = -N, \ldots, N\}, \tag{2.24}$$

where d is the sample spacing in the affine parameter, and $N = Ld^{-1}$.

Therefore, our data, which depend on the angles $\{k\Delta\theta\}_{k=0}^{M}$ and the affine parameters $\{jd\}_{j=-N}^{N}$, are

$$\{\mathcal{R}\mu(jd, k\Delta\theta) : j = -N, \ldots, N,\ k = 0, \ldots, M\}. \tag{2.25}$$

As the first step in the derivation of a reconstruction algorithm, we assume that we can

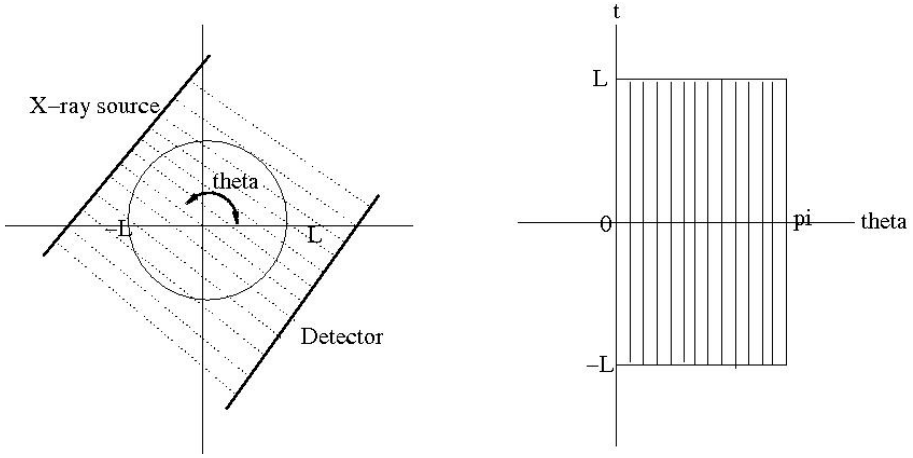


Figure 2.7: A parallel-beam scanner (on the left) and the sample space (on the right).

measure *all* the data from a finite set of equally spaced angles. That is, we assume our samples are extended to

$$\{\mathcal{R}\mu(t, k\Delta\theta) : t \in [-L, L], k = 0, \ldots, M\}. \tag{2.26}$$

The samples (2.26) can be used to compute approximations to samples of the 2-D Fourier transform of μ,

$$\widehat{\mu}(\mathbf{r}_j \boldsymbol{\omega}(k\Delta\theta)), \qquad \text{where } \mathbf{r}_j \in \{0, \pm\eta, \pm 2\eta, \ldots, \pm N\eta\}, \text{with } \eta = \frac{1}{N}\frac{\pi}{d} = \frac{\pi}{L}. \tag{2.27}$$

To accomplish this, we use the Central slice theorem (Theorem 2.2.1):

$$\widehat{\mu}(r\boldsymbol{\omega}(k\Delta\theta)) = \widetilde{\mathcal{R}\boldsymbol{\mu}}(r, k\Delta\theta) = \int_{-\infty}^{\infty} \mathcal{R}\mu(t, k\Delta\theta)e^{-irt}\, dt. \tag{2.28}$$

As mentioned earlier, the r integral in (2.16) is often interpreted as a linear, shift invariant filter, i.e.,

$$\mathfrak{g}\mathcal{R}\mu(t,\theta) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \widetilde{\mathcal{R}\mu}(r,\theta)e^{irt}\,|r|\; dr. \tag{2.29}$$

Therefore, we can write the Radon inversion formula (2.16), also known as the filtered back-projection formula, as

$$\mu(\mathbf{x}) = \frac{1}{2\pi}\int_{0}^{\pi} \mathfrak{g}\mathcal{R}\mu(\langle \mathbf{x}, \boldsymbol{\omega}(\theta)\rangle, \theta)\, d\theta. \tag{2.30}$$

As (2.16) suggests, by using the 2-D Fourier inversion formula and a Riemann sum in the angular direction, we get an approximate value for μ:

$$\begin{aligned} \mu(\mathbf{x}) &\approx \frac{1}{4\pi(M+1)}\sum_{k=0}^{M}\int_{-\infty}^{\infty} \widehat{\mu}(r\boldsymbol{\omega}(k\Delta\theta))e^{ir\langle(\mathbf{x}),\boldsymbol{\omega}(k\Delta\theta)\rangle}|r|\, dr & (2.31)\\ &\approx \frac{1}{2(M+1)}\sum_{k=0}^{M} \mathfrak{g}\mathcal{R}\mu(\langle \mathbf{x}, \boldsymbol{\omega}(k\Delta\theta)\rangle, k\Delta\theta). & (2.32)\end{aligned}$$

In practice, the approximation to the transfer function $\mathfrak{g}$ is chosen to be an approximation to filtering with $|r|$. We denote the approximate transfer function by $\widehat{g}$ defined by (2.19). To develop a reconstruction algorithm for the attenuation coefficient, we approximate the integrals in (2.16) using a quadrature formula, in both the spatial and the frequency domain. Then, using the actual measurements (2.25), we can compute samples of the approximate Fourier transform of μ. More specifically, when the data is collected and filtered, we compute an approximation to the image by using a Riemann sum approximation to the back-projection and find an approximation to the 2-D attenuation

coefficient μ at $\mathbf{x}$ by

$$\mu_{\text{approx}}(\mathbf{x}) = \frac{d}{2(M+1)} \sum_{k=0}^{M} \sum_{j=-N}^{N} \widetilde{\mathcal{R}\mu}(jd, k\Delta\theta)\widehat{g}(\langle \mathbf{x}, \boldsymbol{\omega}(k\Delta\theta)\rangle - jd). \tag{2.33}$$

As explained, the projection data is sampled in polar coordinates, (t, θ). Therefore, after the outer summation in (2.33) (in the continuous case, the outer integral), we use an interpolation to deliver the computed filtered back-projection values to corresponding pixels in the cartesian coordinates.

2.5 Matrix representation of filtered back-projection

In this section we show how to represent each step of the filtered back-projection algorithm in matrix form. This representation helps establish a relation between the iterative methods, in particular ART, and FBP. The iterative methods are discussed in Chapter 3. It also helps us develop the matrix-free polyenergetic iterative reconstruction algorithm in Chapter 5.

The matrix representation of FBP can have other applications as well. For example, it can help us analyze and compare the complexity of the direct and iterative methods. It may also help us find a way to use FBP in an iterative method, e.g., as a preconditioner, to make the computation more efficient.

The input to the FBP algorithm is the Radon transform of the projection data, $\mathbf{P}$, which is also known as the sinogram, as noted before. That is, $\mathbf{P} = [\mathcal{R}\mu(x, y)](t, \theta)$. This matrix $\mathbf{P}$ is of size $N_t \times N_\theta$, where N_θ is the number of projection angles, and N_t is the number of affine parameters, t. It is often convenient to reshape the matrix $\mathbf{P}$ into a 1-D vector $\mathbf{b}$ in column-major order of size $N_{\text{proj}} = N_t \times N_\theta$ and use $\mathbf{b}$ as the input to FBP.

One of the approaches for reconstructing an object is to run the projections back through the image. That is, we filter the projection data (either in the spatial domain or in the frequency domain), and then smear the current filtered projection over the entire

image. Intuitively, this corresponds to an interpolation process from the available set of projections. Different interpolation methods, such as nearest neighbor, linear, cubic, or spline, can be applied. We give an overview of the algorithm. Each step is described in detail in the next sections.

- The fast Fourier transform of $\mathbf{b}$ is calculated:

$$\mathbf{Fb}. \tag{2.34}$$

  This step computes an approximation to $\widetilde{\mathcal{R}\mu}$ (see Equation (2.15)). Note that, we filter the projection data in the frequency domain.

- To filter in the frequency domain, we use the diagonal matrix $\mathbf{D}^{\widehat{\mathbf{g}}}$:

$$\mathbf{D}^{\widehat{\mathbf{g}}}\mathbf{Fb}. \tag{2.35}$$

  This step corresponds to filtering $\widetilde{\mathcal{R}\mu}$ by the radial (i.e., inner) integral in (2.16).

- After filtering in the frequency domain, we use the inverse FFT to transform the filtered projections back to the spatial domain:

$$\mathbf{F}^{(-1)}\mathbf{D}^{\widehat{\mathbf{g}}}\mathbf{Fb}. \tag{2.36}$$

- After computing the filtered sinogram, we perform the back-projection (outer summation in (2.33)) using matrix $\mathbf{E}$:

$$\mathbf{EF}^{(-1)}\mathbf{D}^{\widehat{\mathbf{g}}}\mathbf{Fb}. \tag{2.37}$$

- For a given projection angle, find the interpolation matrix, $\mathbf{J}$, such as *Nearest Neighbor*, to assign the filtered back-projection values to the corresponding pixels.

That is, the image, $\boldsymbol{\mu}$ can be reconstructed by using a series of matrix-vector products:

$$\boldsymbol{\mu} \approx \mathbf{JEF}^{(-1)}\mathbf{D}^{\widehat{\mathbf{g}}}\mathbf{Fb}. \tag{2.38}$$

In the next subsections, we first introduce the Kronecker product in Section 2.5.1. Then we describe the steps outlined above in more detail.

2.5.1 Kronecker product

In this subsection, we introduce the *Kronecker product* to provide additional details for the operations described in Section 2.5. The Kronecker product is an operation on two matrices of arbitrary size resulting in a block matrix. If matrix $\mathbf{A}$ is of size $M \times N$ and matrix $\mathbf{B}$ is of size $P \times Q$, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \otimes \mathbf{B}$, is of size $MP \times NQ$, and is defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}. \tag{2.39}$$

2.5.2 Fourier transform matrix

To filter the projection data in the frequency domain, rather than the spatial domain, we use FFT to transform the projections to the frequency domain. After filtering (see Section 2.4.1), we transform the filtered projection-back to the spatial domain by using the inverse FFT operator explained in Section 2.5.4.

Define $\mathbf{F}_{N_t} = [\mathbf{F}_{jk}]$

$$\mathbf{F}_{jk} = \exp(-2\pi(j-1)(k-1)i/N_t), \tag{2.40}$$

where $j, k = 1, \ldots, N_t$. Then the matrix $\mathbf{F}$ in (2.38) is

$$\mathbf{F} = \mathbf{I}_{N_\theta} \otimes \mathbf{F}_{N_t}, \tag{2.41}$$

which is a block matrix of size $N_t N_\theta \times N_t N_\theta$ having $N_\theta \times N_\theta$ blocks each of size $N_t \times N_t$.

2.5.3 Filtering matrix

After applying the Fourier transform to the projection data, we filter the data in the frequency domain. The filtering matrix $\mathbf{D}^{\widehat{\mathbf{g}}}$ is the same size as $\mathbf{F}$.

2.5.4 Inverse Fourier transform matrix

After filtering, we transform the filtered data back to the spatial domain by applying the inverse Fourier transform. Define $\mathbf{F}_{N_t}^{(-1)} = \left[\mathbf{F}_{jk}^{(-1)}\right]$

$$\left[\mathbf{F}_{jk}^{(-1)}\right] = \frac{1}{N_t} \exp(2\pi(j-1)(k-1)i/N_t), \tag{2.42}$$

where $j, k = 1, \ldots, N_t$. Then let

$$\mathbf{F}^{(-1)} = \mathbf{I}_{N_\theta} \otimes \mathbf{F}_{N_t}^{(-1)}. \tag{2.43}$$

2.5.5 Back-projection – Interpolation

So far we have the values of the filtered sinogram data, $\mathfrak{g}\mathcal{R}\mu(\mathbf{t}_l, \theta_k)$. In the next step, we perform the back-projection by multiplying these values by the matrix $\mathbf{E}$, which is equivalent to the outer summation in (2.33). Then we use an interpolation to assign the filtered back-projection values to the corresponding pixels (x_n, y_m), where $n = 1, \ldots, N_x$, $m = 1, \ldots, N_y$. Note that $\mathbf{E}$ has the same size as $\mathbf{F}$ and $\mathbf{D}^{\widehat{\mathbf{g}}}$.

We use Nearest Neighbor interpolation to illustrate the interpolation process. Given each pixel (x_n, y_m), where $n = 1, \ldots, N_x$, $m = 1, \ldots, N_y$, $N_{\text{grid}} = N_x \times N_y$ and each projection angle θ_k, $k = 1 \ldots N_\theta$, we set

$$\bar{\mathbf{t}}_{knm} = x_n \cos\theta_k + y_m \sin\theta_k. \tag{2.44}$$

Using these $\bar{\mathbf{t}}_{knm}$ values, we implement Nearest Neighbour interpolation as a matrix-vector multiply, as indicated in (2.38), by forming the following interpolation matrix, $\mathbf{J}_{\theta_k}$, of size $(N_x N_y) \times N_t$, for each projection angle θ_k.

$$\mathbf{J}_{\theta_k} = \begin{bmatrix} \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{k11}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{k11}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{k11}) \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{k21}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{k21}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{k21}) \\ \vdots & \vdots & \vdots & \vdots \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{kN_x1}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{kN_x1}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{kN_x1}) \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{k12}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{k12}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{k12}) \\ \vdots & \vdots & \vdots & \vdots \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{kN_x2}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{kN_x2}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{kN_x2}) \\ \vdots & \ddots & \ddots & \vdots \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{k1N_y}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{k1N_y}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2})}(\bar{t}_{k1N_y}) \\ \vdots & \vdots & \vdots & \vdots \\ \chi_{[\mathbf{t}_{1/2},\mathbf{t}_{3/2})}(\bar{t}_{kN_xN_y}) & \chi_{[\mathbf{t}_{3/2},\mathbf{t}_{5/2})}(\bar{t}_{kN_xN_y}) & \cdots & \chi_{[\mathbf{t}_{N_t-1/2},\mathbf{t}_{N_t+1/2}]}(\bar{t}_{kN_xN_y}) \end{bmatrix}_{(NxNy)\times N_t} \tag{2.45}$$

where χ is the characteristic function defined in (2.20), $\mathbf{t}_{1/2} = -\infty$, and $\mathbf{t}_{N_t+1/2} = \infty$. We have N_θ matrix blocks and therefore the final interpolation matrix is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\theta_1} & \mathbf{J}_{\theta_2} & \ldots & \mathbf{J}_{\theta_{N_\theta}} \end{bmatrix}. \tag{2.46}$$

We can also use other interpolation schemes, such as linear interpolation, spline interpolation or the Piecewise *Cubic Hermite Interpolating Polynomial* (CHIP) method. We use the Matlab implementation of the CHIP method (called `pchip`) in Section 5.6. CHIP is a third-degree spline method with each polynomial of the spline in Hermite form. For more information about this method refer to [24] and [33].

Chapter 3

Iterative Reconstruction Algorithms

3.1 Introduction

Reconstruction algorithms in CT fall into two categories: direct (e.g., filtered back-projection (FBP)) or iterative (e.g., algebraic reconstruction techniques (ART) and statistical image reconstruction techniques (SIRT)). FBP, the most commonly used reconstruction algorithm in CT, is based on the Central slice theorem as explained in Chapter 2. In recent years, the limits of FBP have been challenged by the progress toward scanning geometries, such as cone-beam and multi-slice helical CT. Therefore, there has been more interest in iterative methods for CT image reconstruction.

Iterative methods can have several advantages over direct methods. These methods can incorporate some prior knowledge, including system geometry, detector response, object constraints, and they also permit modeling data noise. Also, an assumption underlying FBP is that x-ray sources are monoenergetic; in practice, there is a nonuniform distribution of photons of different wavelengths (hence, different energies) that leads to a phenomenon physicists call beam hardening. Different methods to compensate for the effects of beam hardening have been proposed, such as prefiltering [9, 40]; post-reconstruction [32, 62]; and incorporating a polyenergetic acquisition

model [17, 18, 37, 61]. Some iterative methods, such as statistical image reconstruction techniques (SIRT), can model polyenergetic x-ray sources and thus account for beam hardening in the reconstruction. They use a statistical model, such as a likelihood-based approach, to estimate the attenuation coefficient.

On the other hand, the main disadvantage of these iterative reconstruction methods is their high computation time [6, 17, 18, 19, 28, 37]. In this chapter, we review several iterative methods.

3.2 Algebraic reconstruction techniques (ART)

The algebraic reconstruction technique (ART) is a completely different approach for image reconstruction from those described in Section 2.4. This method was first proposed in 1970 [25]. As for FBP, we assume that the cross-section of the object to be imaged consists of an array of unknowns. Therefore, the reconstruction problem can be formulated as a system of linear equations and then an iterative methods is applied to find the unknown attenuation coefficient. Conceptually this method is simpler than filtered back-projection (FBP). Also, FBP is less adaptable than ART to missing data and partial occlusion effects. On the other hand, for comparable results in medical applications, algebraic techniques are computationally more expensive (in terms of both storage and computation time).

In the following sections, we briefly describe the mathematical formulation of ART and its properties. The presentation adopted in this section is drawn largely from [19].

3.2.1 Mathematical formulation

We consider the problem of reconstructing a 2-D attenuation coefficient $\mu = \mu(x, y)$. The first step in an algebraic reconstruction technique is to choose a finite collection of basis

functions

$$\{\mathcal{B}_1(u,v),\ldots,\mathcal{B}_J(u,v)\}. \tag{3.1}$$

It is assumed that the attenuation coefficient can be approximated by functions in the linear span of the basis functions. Now considering a two-dimensional slice of attenuation coefficient, μ, our goal is to find the coefficients $\{\mathbf{m}_j\}$ that minimize the difference

$$\left\| \mu - \sum_{j=1}^{J} \mathbf{m}_j \mathcal{B}_j \right\| \tag{3.2}$$

in some suitable function norm to be chosen. We typically use localized basis functions, each of which has a bounded support in a small set. The *pixel basis*, described below, is an example of such a basis.

3.2.2 Pixel basis

Suppose that the domain of μ is the square $[-1,1]\times[-1,1]$ and this square is uniformly divided into a $K\times K$ grid . We label the subsquares sequentially from left to right and then bottom to top, as illustrated in Figure 3.1. The elements of the $K\times K$ pixel basis are defined by

$$\mathcal{B}_j^K(u,v) = \begin{cases} 1 & \text{if} \qquad (u,v)\in j\text{th-square}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

Now if $\mathbf{m}_j$ is the average value of μ in the jth-square, then an approximation of μ in terms of the pixel basis is

$$\bar{\mu}^K = \sum_{j=1}^{J} \mathbf{m}_j \mathcal{B}_j^K, \tag{3.4}$$

where $J=K^2$. If μ is continuous on $[-1,1]^2$ with bounded support, the sequence $\{\bar{\mu}^K\}$ converges uniformly to μ as $K\to\infty$, i.e., the image defined by $\bar{\mu}^K$ converges to the image represented by the function μ as $K\to\infty$.

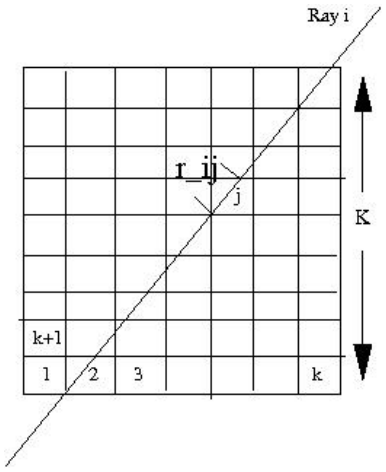


Figure 3.1: Geometric construction (using one-dimensional rays) of elements of the measurement matrix in the pixel basis, where $\mathbf{r}_{ij}$ is the length of the intersection of the ith ray with the jth pixel.

The Radon transform is linear and therefore

$$\mathcal{R}\bar{\mu}^K = \sum_{j=1}^{J} \mathbf{m}_j \mathcal{R}\mathcal{B}_j^K. \tag{3.5}$$

It is easy to compute $\mathcal{R}\bar{\mu}^K$, since it is easy to compute each $\mathcal{R}\mathcal{B}_j^K$ on the right side of (3.5). This is an advantage since the measurements are modeled as samples of $\mathcal{R}\mu$.

3.2.3 Reconstruction model

A *measurement matrix* models the result of applying the measurement process to the basis functions. Assume that $\{\mathcal{B}_j\}$ is a localized basis, but not necessarily the pixel basis. The samples are labeled sequentially by $i \in \{1, \ldots, I\}$, with $\mathcal{R}\mu$ sampled at lines $l_{\mathbf{t}_i, \boldsymbol{\omega}_i}$, where

$$(\mathbf{t}_i, \boldsymbol{\omega}_i) \in \{(\mathbf{t}_1, \boldsymbol{\omega}_1), \ldots, (\mathbf{t}_I, \boldsymbol{\omega}_I), i = 1, \ldots, I\}. \tag{3.6}$$

One way to define the measurement matrix is as the line integrals

$$\mathbf{r}_{ij} = \mathcal{R}\mathcal{B}_j(\mathbf{t}_i, \boldsymbol{\omega}_i), \qquad i = 1, \ldots, I,\ j = 1 \ldots, J. \tag{3.7}$$

Then the entries of the vector of measurements are

$$\mathbf{b}_i = \mathcal{R}\mu(\mathbf{t}_i, \boldsymbol{\omega}_i), \qquad i = 1, \ldots, I. \tag{3.8}$$

The reconstruction problem can now be formulated as a system of I linear equations in J unknowns:

$$\sum_{j=1}^{J} \mathbf{r}_{ij}\mathbf{m}_j = \mathbf{b}_i, \qquad i = 1 \ldots, I, \tag{3.9}$$

or equivalently, in matrix form, as

$$\mathbf{rm} = \mathbf{b}. \tag{3.10}$$

For a geometric interpretation of the $\mathbf{r}_{ij}$s, see Figure 3.1 and refer to [19].

Following the survey [10] of iterative algorithms, the variables in these equations are interpreted as follows: the unknown entries of the vector $\mathbf{m}$ often represent intensity levels of beam attenuation in transmission tomography, or of radionuclide concentration in emission tomography. The entries of the vector of measurements $\mathbf{b}$ are typically counts of detected photons. The vector $\mathbf{b}$ consists of exactly the same sinogram data that is used in FBP. The measurement matrix describes a relationship between the various pixels and the various detectors. That is, it describes the scanning process whereby the information about the unknown function is translated into measured data.

The size of these systems of equations is usually large making ART methods computationally expensive. To give an indication of the extent of the difficulty, we quote a passage from [19, p. 495]:

> "If the square is divided into $J = 128 \times 128 \simeq 16,000$ subsquares, then, using the pixel basis, there are 16,000 unknowns. A reasonable number of measurements is 150 samples of the Radon transform at each of 128 equally spaced angles, so that $I \simeq 19,000$. That gives a $19,000 \times 16,000$ system of equations. Even today it is not practical to solve a system of this size."

A practical ART method has to maintain a balance between the computational load of solving (3.10) and the inconsistencies that result from a crude grid. To solve (3.10) effectively, we need to use relatively sophisticated techniques from numerical linear algebra.

3.2.4 Kaczmarz's method

The principle method used in ART is derived from *Kaczmarz's Method*, often also called the method of projections. In this method, like other iterative methods, instead of solving (3.10) directly, we use an algorithm that defines a sequence $\mathbf{m}^{(k)}$ of vectors that get closer and closer to a solution (or approximate solution) with respect to a norm.

We start with an initial guess $\mathbf{m}^{(0)}$. Then subsequent iterates are computed by projecting orthogonally onto the hyperplanes defined by (3.9). To be more specific, the new iterates are computed by

$$\mathbf{m}^{(k+1)} = \mathbf{m}^{(k)} + \frac{\mathbf{b}_i - \langle \mathbf{r}_i, \mathbf{m}^{(k)} \rangle}{\langle \mathbf{r}_i, \mathbf{r}_i \rangle} \mathbf{r}_i, \tag{3.11}$$

where $\mathbf{r}_i$ is a vector containing the ith row of the matrix $\mathbf{r}$. Note that $\langle \mathbf{r}_i, \mathbf{m}^{(k+1)} \rangle = \mathbf{b}_i$, i.e., $\mathbf{m}^{(k+1)}$ satisfies the ith row.

Typically, the iteration is stopped when $||\mathbf{r}\mathbf{m}^{(k)} - \mathbf{b}||_2 \leq TOL$ for some user-defined tolerance, TOL. However, in medical applications, only a few complete iterations are normally used. In [19, 42], it is claimed that the quality of the reconstructed image often improves for a few iterates but then begins to deteriorate.

We usually have noisy data. To reduce the effect of the noise and to accelerate the process, some kind of *smoothing*, also known as *relaxation*, is frequently incorporated into the ART algorithm, extending it to

$$\mathbf{m}^{(k+1)} = \mathbf{m}^{(k)} + \lambda^{(k)} \frac{\mathbf{b}_i - \langle \mathbf{r}_i, \mathbf{m}^{(k)} \rangle}{\langle \mathbf{r}_i, \mathbf{r}_i \rangle} \mathbf{r}_i, \tag{3.12}$$

where $\lambda^{(k)}$ is the relaxation parameter at the kth iteration. In [30], it is shown that if we carefully adjust the order in which the collected data are accessed during the reconstruction procedure, and choose the relaxation parameters judiciously, ART can produce high-quality reconstructions efficiently. Several books and papers, such as [10], [19] and [34], have discussed relaxation parameters. For a detailed description of different variants of Kaczmarz's iteration, refer to [2, 3, 29, 31].

3.3 Statistical image reconstruction techniques (SIRT)

Statistical image reconstruction techniques (SIRT) are based on modeling assumptions that incorporate the stochastic nature of physical measurements. As with other CT reconstruction algorithms, the basic idea in SIRT is to find the distribution of the energy-dependent attenuation coefficient μ given the measurements $\mathbf{b}$. In FBP, usually monoenergetic x-ray beams are assumed, and therefore the issue of beam hardening (see Section 3.3.2) is not considered. Statistical methods allow us to assume polyenergetic sources, and thereby reduce the negative affects of beam hardening artifacts.

In statistical methods, a physical, statistical acquisition model is assumed first. Then a statistical model, such as a likelihood-based approach, is used to estimate the attenuation coefficient. In practice, the data is often noisy and, consequently, the likelihood estimates give noisy reconstructions. Therefore, in some algorithms the next step would be regularization. At the end, the estimation found is optimized, by applying an iterative method.

3.3.1 Polyenergetic physical model

In Section 2.1, we describe a physical model based on a monoenergetic source. However, in reality x-rays are polyenergetic. In this section, we describe a physical model, used in [13, 17, 18, 37, 54], that accounts for a polyenergetic x-ray source spectrum. Therefore,

the linear attenuation coefficient depends on both the spatial coordinates $\mathbf{x}$ (in 2-D, $\mathbf{x} = (x, y)$) and the beam energy ε. If we use the same variables as in Section 2.1, then, for a ray $l_{t,\theta}$, the projection measurement $P_{t,\theta}$ is given by

$$P_{t,\theta} = \int_{\varepsilon_{min}}^{\varepsilon_{max}} P_0(\varepsilon) \exp\left[-\int_{l_{t,\theta}} \mu(\mathbf{x}, \varepsilon)\, ds \right] d\varepsilon. \tag{3.13}$$

Clearly, if we assume a monoenergetic source, this equation reduces to (2.2). Different discretizations may be applied to approximate the integrals in (3.13).

3.3.2 Beam hardening

In practice, x-ray beams produced in CT scanners are polyenergetic with a relatively wide energy spectrum. Moreover the attenuation coefficients are energy-dependent. Low energy x-rays, which are more easily attenuated, are called soft x-rays. On the other hand, high energy x-rays, which are more penetrating, are referred to as hard. The beam hardening phenomenon is the process of increasing the average energy level of an x-ray beam, or "hardening" of the x-ray beam, as it passes through the scanned object (see Figure 3.2). This happens because, as a polyenergetic beam passes through an object, it loses the lower-energy parts of its spectrum more rapidly than the higher-energy parts of the spectrum,, since the lower-energy parts are more easily attenuated.

The degree to which a given x-ray beam is hardened in passing through matter depends on both the initial x-ray spectrum and the composition of the material or tissue traversed. However, for any fixed initial x-ray spectrum and tissue type, the process of beam hardening represents a monotonic increase in beam hardness as a function of distance. In other words, the attenuation coefficient depends on the thickness of material traversed. This effect causes "beam hardening artifacts" in CT images. Figure 3.3 shows the effect of beam hardening on images. Notice in particular the false line integrals

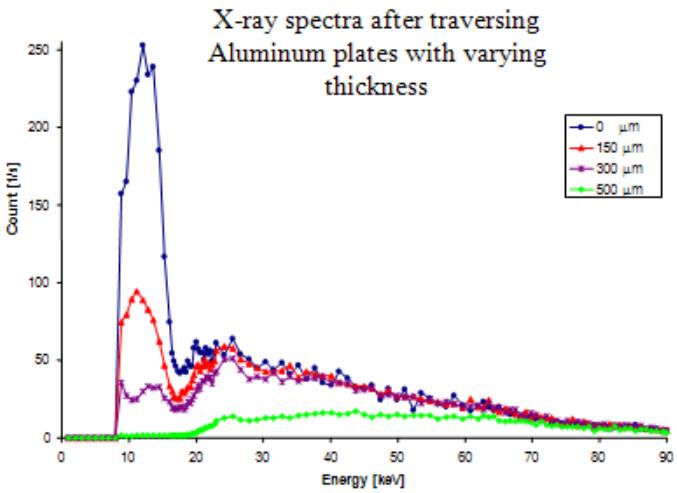


Figure 3.2: This graph shows that the average energy level of an x-ray beam increases when the low-energy photons are filtered out. This phenomenon is known as beam hardening. This picture is taken from `http://aladdin.utef.cvut.cz/ofat/Methods/BeamHardening/BeamHardening.html`

between the inner ellipses.

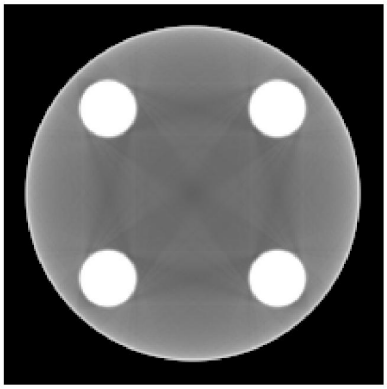


Figure 3.3: This image contains beam hardening artifacts. The false line integrals between the inner ellipses are clear.

Generally, beam hardening leads to a reduction in the reconstructed attenuation coefficient [5]. The computed values in soft tissues are depressed nonuniformly, and the hard tissues, such as bones, generate dark streaks. Also, bone areas can spill over into

soft tissue, leading to a perceived increase in the attenuation coefficient [17, 18]. Different techniques have been proposed to reduce the incidence of beam hardening artifacts in x-ray CT reconstructions.

- **Pre-filtering** A physical device is used to ensure that the x-ray beams used are closer to being truly monoenergetic [9, 40], making the assumption of monoenergetic x-ray beams more reasonable.

- **Post-reconstruction** This is a standard post-processing method proposed by Joseph and Spital in 1978 [32]. This method relies on assumptions about the material characteristics to provide corrections to the measured sinogram data. The reconstruction is done in two stages: an approximate material distribution is assumed at first, and the corresponding beam hardening artifacts are then reduced.

- **Incorporating a polyenergetic acquisition model** Statistical image reconstruction techniques (SIRT) for x-ray CT can be developed based on physical models that account for polyenergetic sources. In this case, since the reconstruction algorithm is built upon a polyenergetic acquisition model, the beam hardening phenomenon is taken into account.

3.3.3 Incorporating a polyenergetic acquisition model

The derivation of FBP (as well as many other reconstruction algorithms for x-ray CT) is strongly dependent on the assumption that the radioactive source used produces monoenergetic x-ray beams that travel in straight lines. Consequently, in such algorithms, beam hardening is neglected. On the other hand, in image reconstruction methods that are based on a polyenergetic physical model (3.13), such as methods discussed in [13, 17, 18, 37, 54], beam hardening is taken into account.

The majority of the statistical image reconstruction algorithms are iterative and likelihood-based [16, 57]. The basic idea of most of these methods is to find the distribu-

tion of μ by maximizing a likelihood of some objective function based on the polyenergetic physical model (3.13). These techniques usually make some prior assumptions about the physical properties of the scanned object and therefore, simplify the problem. For example, in [17, 18], it is assumed that the scanned object consists of a given number of nonoverlapping materials, such as soft tissue and bone. To simplify the problem more, it is also assumed that μ is the product of its unknown density and a known energy-dependent mass attenuation coefficient. Then, a penalized-likelihood function for the polyenergetic model (3.13) is formulated. Finally, an iterative method using some surrogate functions is used for estimating the unknown densities in each voxel [20, 47, 48].

In Chapters 4 and 5, we introduce and discuss a new iterative polyenergetic reconstruction method that is based on the same polyenergetic model (3.13). However, one of the advantages of our new method is that we do not assume any prior knowledge to simplify the problem. We discretize (3.13) directly and find the attenuation coefficient by applying different optimization techniques.

Chapter 4

Polyenergetic Nonlinear Iterative Method

We discuss the development of a non-statistical iterative reconstruction method that is based on the polyenergetic model described in Section 3.3.1. We discretize the polyenergetic model (3.13) directly with respect to different energy levels and different projections. To this end, we have studied different solvers and nonlinear unconstrained optimization methods, such as a Newton-like method and an extension of the Levenberg-Marquardt-Fletcher algorithm. We solve the reconstruction problem by incorporating such iterative optimization frameworks and present numerical results.

4.1 More about the polyenergetic physical model

In Section 2.1, we reviewed a physical model based on a monoenergetic source. In reality, x-rays are polyenergetic and hence, a polyenergetic model can lead to results with better image quality. In Section 3.3.1, we briefly discussed such models, which, to date, have mostly been used in statistical image reconstruction techniques. Here, we continue the discussion of a physical model, used in [13, 17, 18, 37, 54], that accounts for a polyenergetic x-ray source spectrum. We use this model in the following sections of this chapter and

Chapter 5.

Assume that, for any line $l_{t,\theta}$, we have the projection measurement $P_{t,\theta}$. If μ is the linear attenuation coefficient which depends both on the spatial coordinates, $\mathbf{x}$ (in 2-D $\mathbf{x} = (x, y)$) and the beam energy, ε, then, as noted in Section 3.3.1, the polyenergetic model can be written as

$$P_{t,\theta} = \int_{\varepsilon_{min}}^{\varepsilon_{max}} P_0(\varepsilon) \exp\left[-\int_{l_{t,\theta}} \mu(\mathbf{x}, \varepsilon)\, ds \right] d\varepsilon. \tag{4.1}$$

In this polyenergetic model, the measurements $P_{t,\theta}$ are nonlinearly dependent on the energy-dependent attenuation coefficient. Note that, if we assume a monoenergetic source, then $\mu(\mathbf{x}, \varepsilon) = \mu(\mathbf{x})$ (since we have one energy level). Hence, (4.1) can be transformed to (2.2). Different discretization methods can be applied to the two integrals in (4.1). This is discussed in more detail in Section 4.2.

4.2 Discretization

4.2.1 Discretizing the outer integral

We discretize the outer integral with respect to ε, representing energy levels, by applying a quadrature rule, such as a Gaussian quadrature rule or the composite Trapezoidal rule on a nonuniform mesh. We use several points on the x-ray spectrum curve in Figure 4.1 as our quadrature nodes. The x-ray spectrum is photon count versus energy value. Standard data for these curves can be found at the National Institute of Standards and Technology (NIST) x-ray mass-attenuation database. In Figure 4.1, we show a typical x-ray spectrum with different maximum energies.

For our problem, we use N_ε quadrature nodes and weights, representing N_ε different energy levels and use them to discretize the outer integral in (4.1) with respect to ε. That

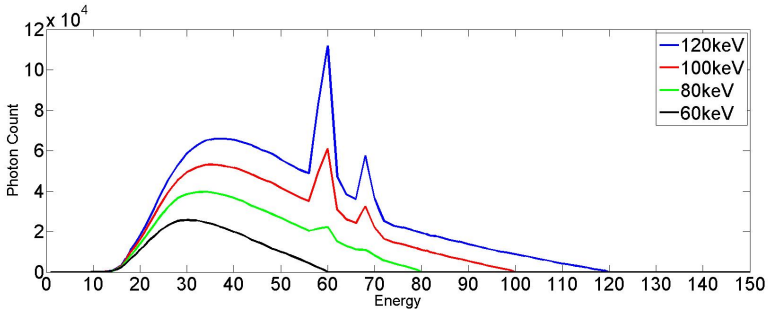


Figure 4.1: Typical x-ray spectrum with different maximum energies.

is, we assume that the projection measurement $P_{t,\theta}$ satisfies the discretized equation

$$\mathbf{P}_{t,\theta} = \sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp\left(- \int_{l_{t,\theta}} \mu(\mathbf{x}, \boldsymbol{\varepsilon}_h) ds \right), \tag{4.2}$$

rather than the corresponding continuous equation (4.1), where in (4.2) $\{\mathbf{w}_h\}_{h=1}^{N_\varepsilon}$ is the set of quadrature weights and $\{\boldsymbol{\varepsilon}_h\}_{h=1}^{N_\varepsilon}$ is the set of quadrature nodes in the range $[\varepsilon_{\min}, \varepsilon_{\max}]$. For smooth curves, such as the low energy curves in Figure 4.1, we use a Gaussian quadrature rule. However, for non-smooth curves with high peaks, we use the composite Trapezoidal rule on a nonuniform mesh. We note that the discretized version of (4.2) is also used to construct our mathematical phantom sinogram data, as explained in more detail in Section 4.7.

4.2.2 Discretizing the inner integral

We consider two different approaches for discretizing the inner integral in (4.1). In the first approach, we solve for the Radon transform of the attenuation coefficient and then use any standard CT reconstruction method, such as FBP, to solve for the attenuation coefficient. In the second approach, we solve for the attenuation coefficient directly.

Solve for the Radon transform of the attenuation coefficient

Suppose that we have measurement data $\mathbf{P}_{t_m,\theta_n}$, $m = 1, \ldots, N_t$ and $n = 1, \ldots, N_\theta$, for appropriately chosen discrete values $\mathbf{t}_m, \theta_n$. Our goal is to solve the discretized version of (4.2)

$$\mathbf{P}_{t_m,\theta_n} = \sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp\left(- \int\limits_{l_{t_m,\theta_n}} \mu(\mathbf{x}, \boldsymbol{\varepsilon}_h) ds \right) \tag{4.3}$$

for an approximation to the attenuation coefficient $\mu(\mathbf{x}, \boldsymbol{\varepsilon}_h)$. In the first approach, we replace the Radon transform $\int\limits_{l_{t_m,\theta_n}} \mu(\mathbf{x}, \boldsymbol{\varepsilon}_h) ds$ in (4.3) by the real variable $\alpha_{t_m,\theta_n,h}$. Then we solve the system of underdetermined equations

$$\mathbf{P}_{t_m,\theta_n} = \sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp\left(-\alpha_{t_m,\theta_n,h}\right) \tag{4.4}$$

for the variables $\alpha_{t_m,\theta_n,h}$. Next, for each h, $h = 1, \ldots, N_\varepsilon$, we fix h and solve the standard CT reconstruction

$$\int\limits_{l_{t_m,\theta_n}} \mu(\mathbf{x}, \boldsymbol{\varepsilon}_h) ds = \alpha_{t_m,\theta_n,h}, \tag{4.5}$$

for an approximation to the attenuation coefficient $\mu(\mathbf{x}, \boldsymbol{\varepsilon}_h)$.

The advantage of this approach is that we can easily use highly efficient codes, such as FBP, to solve the system (4.5). However, the system (4.4) that we must solve for the $\alpha_{t_m,\theta_n,h}$ consists of $N_{\text{proj}} = N_t \times N_\theta$ equations and $N_{\text{proj}} N_\varepsilon$ unknowns. So the number of unknowns in the system (4.4) is N_ε times larger than the number of equations. Hence, the system is extremely underdetermined even if N_ε is of moderate size. As a result, we were not able to get this approach to work well. Therefore, we do not use it for our computations reported later in the thesis. However, we mention it here because it does have some appealing features and it may prove useful in the future.

Solve for the attenuation coefficient

Another option for discretizing the inner integral in (4.2) is to discretize the Radon transform and solve for the attenuation coefficient directly. To do this we can either use the ART projection matrix (see Section 3.2) or use the Radon transform to compute the line integral [39]. In the ART approach, computing the projection matrix can be very expensive, in terms of both computing time and the memory required to store it. Hence, we choose to use the Radon transform since there are several efficient algorithms to compute it (see Section 4.2.2) and we do not need to store the associated matrix. Let $\mathbf{R}$ represent the $N_{\text{proj}} \times N_{\text{grid}}$ Radon matrix, where $N_{\text{proj}} = N_t N_\theta$ is the number of affine parameters multiplied by the number of projection angles. Also note that, N_{grid} is the number of pixels (or voxels in 3-D cases) in the reconstructed image. Let $\mathbf{R}_i^T$ be the i-th row of $\mathbf{R}$, for $i = 1, \ldots, N_{\text{proj}}$. Thus, $\mathbf{R}_i^T$ is a N_{grid}-vector. Therefore, the Radon matrix can be written as

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_1^T \\ \mathbf{R}_2^T \\ \vdots \\ \mathbf{R}_{N_{\text{proj}}}^T \end{pmatrix} = [\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{N_{\text{proj}}}]^T. \tag{4.6}$$

Now note that the Radon matrix $\mathbf{R}$ approximates the Radon transform $\mathcal{R}$ in the sense that

$$[\mathbf{R}\boldsymbol{\mu}]_{m,n} \approx [\mathcal{R}\mu](\mathbf{t}_m, \theta_n). \tag{4.7}$$

The μ on the right side of (4.7) denotes the continuous attenuation coefficient; the $\boldsymbol{\mu}$ on the left side represents a discrete approximation of μ. With the discretizations in (4.2) and (4.6), the original equations (4.1) can be approximated by

$$\mathbf{P}_i = \sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h), \tag{4.8}$$

where $i = 1, \ldots, N_{\text{proj}}$, $N_{\text{proj}} = N_t \times N_\theta$ and $\boldsymbol{\mu}_h$ is an approximation to the energy dependent attenuation coefficient $\mu(\mathbf{x}, \boldsymbol{\varepsilon}_h)$.

Discrete Radon transform

In our implementations, we use the MATLAB function `radon` to approximate the Radon transform [39]. In MATLAB, the Radon transform of an image is the sum of the Radon transforms of each individual pixel. The algorithm first divides pixels in the image into four subpixels and projects each subpixel separately, as shown in Figure 4.2. Each subpixel's contribution is proportionally split into the two nearest bins, according to the distance between the projected location and the bin centers. If the subpixel projection hits the center point of a bin, the bin on the axes gets the full value of the subpixel, or one-fourth the value of the pixel. If the subpixel projection hits the border between two bins, the subpixel value is split evenly between the bins. There are other methods for discretizing the Radon transform, but we will not discuss those methods here. In the next

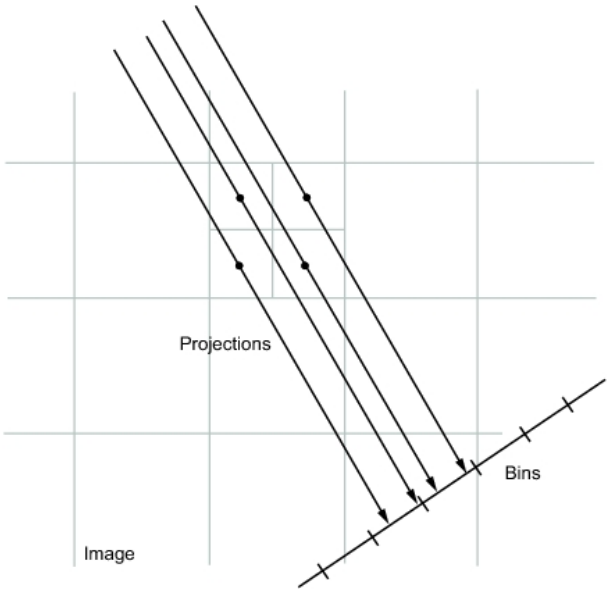


Figure 4.2: The Radon algorithm first divides pixels in the image into four subpixels and projects each subpixel separately. This picture is taken from the MATHWORKS website `http://www.mathworks.com/help/toolbox/images/ref/radon.html`

sections, we discuss how to filter the sinogram data and solve the discretized problem. We can use different optimization methods or solvers to find the unknown energy-dependent attenuation coefficient. This is discussed in more detail in Sections 4.4, 4.5 and 4.6.

4.3 Filtering

For filtering, we apply the method discussed in Section 2.4.1. We remind the reader that we filter the noisy sinogram data as follows:

1. Take the one dimensional FFT of the sinogram data;
2. To extract the desired signals from the raw data, instead of convolving a filter with the sinogram data, multiply the filter (defined in the frequency domain) with the FFT of the sinogram elementwise. In other words, the filtering is performed as a multiplication in the frequency domain instead of a convolution in the spatial domain.
3. Take the inverse Fourier transform of the filtered sinogram in the frequency domain.

We may filter our initial guess and then use our reconstruction algorithm; or filter the result at the end. We discuss some regularization methods, which can replace or complement the filtering stage, in Section 5.5.

4.4 Unconstrained optimization

So far, we have discretized the continuous model and written it as (4.8). The goal of our reconstruction problem, like other reconstruction problems, is to approximate the attenuation coefficient. Therefore, we need to develop a method to find the energy-dependent attenuation coefficient that makes the right side of (4.8) as close as possible to its left side. In order to find such a method, we first have to define an objective function and ascertain its properties.

4.4.1 Objective function

Define a vector-valued nonlinear function $\mathbf{F}(\boldsymbol{\mu})$ with N_{proj} components as follows. For each projection i, $i = 1, \ldots, N_{\text{proj}}$, use the discretized model (4.8) to define

$$\mathbf{F}_i(\boldsymbol{\mu}) = \sum_{h=1}^{N_\varepsilon} \boldsymbol{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h) - \mathbf{P}_i. \tag{4.9}$$

Our goal is to minimize

$$G(\boldsymbol{\mu}) = \|\mathbf{F}(\boldsymbol{\mu})\|_2^2, \tag{4.10}$$

which is a scalar-valued objective function. This is a nonlinear least-squares fitting problem.

Jacobian/Gradient of the objective function

We choose to use an optimization method that uses the Gradient of $G(\boldsymbol{\mu})$ or the Jacobian of $\mathbf{F}(\boldsymbol{\mu})$, e.g., the Levenberg-Marquardt algorithm (see Section 4.6). The Jacobian of $\mathbf{F}(\boldsymbol{\mu})$ is the matrix of partial derivatives $\frac{\partial \mathbf{F}_i(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}}$, where

$$\begin{aligned}
\frac{\partial \mathbf{F}_i(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}} &= \frac{\partial}{\partial \boldsymbol{\mu}_{j,h}} \left\{ \sum_{h^*=1}^{N_\varepsilon} \mathbf{w}_{h^*} \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_{h^*}) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_{h^*}) - \mathbf{P}_i \right\} \\
&= \sum_{h^*=1}^{N_\varepsilon} \mathbf{w}_{h^*} \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_{h^*}) \frac{\partial}{\partial \boldsymbol{\mu}_{j,h}} \left\{ \exp\left(-\mathbf{R}_i^T \boldsymbol{\mu}_{h^*}\right) \right\} \\
&= \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h)(-\mathbf{R}_{i,j}) \left\{ \exp\left(-\mathbf{R}_i^T \boldsymbol{\mu}_h\right) \right\},
\end{aligned} \tag{4.11}$$

for $i = 1, \ldots, N_{\text{proj}}$, $j = 1, \ldots, N_{\text{grid}}$ and $h = 1, \ldots, N_\varepsilon$. Note that, in (4.11), we use h^* as a counter and h is a fixed index for the energy level associated with the derivative. The Jacobian matrix is of size $N_{\text{proj}} \times N_{\text{grid}} N_\varepsilon$. The scalar-valued objective function is

$$G(\boldsymbol{\mu}) = \|\mathbf{F}(\boldsymbol{\mu})\|_2^2 = \sum_{i=1}^{N_{\text{proj}}} \mathbf{F}_i(\boldsymbol{\mu})^2. \tag{4.12}$$

The gradient of $G(\boldsymbol{\mu})$ is the vector of partial derivatives $\left[\frac{\partial \mathbf{G}(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}}\right]$, where

$$\begin{aligned}\frac{\partial G(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}} &= \sum_{i=1}^{N_{\text{proj}}} \frac{\partial}{\partial \boldsymbol{\mu}_{j,h}}\{\mathbf{F}_i^2(\boldsymbol{\mu})\} \\ &= \sum_{i=1}^{N_{\text{proj}}} 2\mathbf{F}_i(\boldsymbol{\mu})\frac{\partial}{\partial \boldsymbol{\mu}_{j,h}}\{\mathbf{F}_i(\boldsymbol{\mu})\}, \end{aligned} \tag{4.13}$$

for $j = 1, \ldots, N_{\text{grid}}$ and $h = 1, \ldots, N_\varepsilon$. We can use the Jacobian of $\mathbf{F}(\boldsymbol{\mu})$ and the Gradient of $G(\boldsymbol{\mu})$ in a variety of different optimization methods to solve our problem.

4.5 Newton's method

The first method we consider is a Newton-like method to solve our nonlinear reconstruction problem. We call it a Newton-like method even though our system is extremely underdetermined. One of the reasons that we choose to experiment with Newton's method is that this method can often converge remarkably quickly, especially if the starting guess is sufficiently close to the desired minimizer. However, if the iteration begins far from the desired minimizer, this method can fail to converge. In our problem, since we use the FBP solution as the initial guess, we have a relatively good starting point. The kth iteration of Newton's method applied to (4.2) is

$$\mathbf{F}'(\boldsymbol{\mu}^{(k)})(\boldsymbol{\mu}^{(k+1)} - \boldsymbol{\mu}^{(k)}) = -\mathbf{F}(\boldsymbol{\mu}^{(k)}), \tag{4.14}$$

or

$$\mathbf{F}'(\boldsymbol{\mu}^{(k)})(\Delta\boldsymbol{\mu}^{(k)}) = -\mathbf{F}(\boldsymbol{\mu}^{(k)}), \tag{4.15}$$

where $\boldsymbol{\mu}^{(k)}$ is the kth approximation to the discrete attenuation coefficient $\boldsymbol{\mu}$; $\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \Delta\boldsymbol{\mu}^{(k)}$; and we compute $\mathbf{F}'(\boldsymbol{\mu})$, the Jacobian of $\mathbf{F}(\boldsymbol{\mu})$, by (4.11).

Now, if we define the multipliers

$$\mathbf{m}_{ih} = \mathbf{w}_h \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h^{(k)}) \tag{4.16}$$

and the right side values

$$\mathbf{Q}_i = -\sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h^{(k)}) + \mathbf{P}_i, \tag{4.17}$$

then we have

$$\sum_{h=1}^{N_\varepsilon} \mathbf{m}_{ih} \mathbf{R}_i^T \Delta \boldsymbol{\mu}_h^{(k)} = \mathbf{Q}_i. \tag{4.18}$$

Although this system is not normally square, we can solve it by applying the method of least squares [7, 44].

Unfortunately, this Newton-like approach does not produce good quality results. In the next sections, we discuss other more promising methods for solving the reconstruction problem.

4.6 Levenberg-Marquardt-Fletcher algorithm

The Levenberg-Marquardt (LM) algorithm is one of the most widely used algorithms for nonlinear least squares problems. The LM algorithm is a combination of gradient descent and Gauss-Newton iteration [44]. This method can also be adapted to the trust region framework [44].

Ignoring our specific problem for now, we describe the LM algorithm and the Levenberg-Marquardt-Fletcher extension in a general setting. Assume we want to minimize the objective function

$$\|\mathbf{r}(\mathbf{x})\|_2^2 = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}), \tag{4.19}$$

where $\mathbf{r}(\mathbf{x}) = \mathbf{b} - \mathbf{f}(\mathbf{x})$ is the residual for some nonlinear system of equations. Our goal

is to find a sequence of iterates $\mathbf{x}^{(k)}$ such that $\mathbf{x}^{(k)} \longrightarrow \mathbf{x}^*$, where $\mathbf{x}^*$ minimizes $\|\mathbf{r}(\mathbf{x})\|_2^2$, whence $\mathbf{f}(\mathbf{x}^*) \approx \mathbf{b}$. Let

$$\Delta\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}. \tag{4.20}$$

If the residuals $\mathbf{r}(\mathbf{x})$ are smooth, then

$$\mathbf{r}^{(k+1)} = \mathbf{r}(\mathbf{x}^{(k+1)}) = \mathbf{r}^{(k)} + \mathbf{A}^{(k)}\Delta\mathbf{x}^{(k)} + \dots , \tag{4.21}$$

where the elements of the Jacobian matrix, $\mathbf{A}^{(k)}$, are

$$\mathbf{A}^{(k)}_{m,n} = \frac{\partial \mathbf{r}_m(\mathbf{x}^{(k)})}{\partial x_n}. \tag{4.22}$$

If we multiply (4.21) by $\mathbf{A}^{(k)^T}$ on the left and rearrange the terms of the equation, we find

$$\mathbf{A^{(k)}}^T\mathbf{A^{(k)}}\Delta\mathbf{x}^k - \mathbf{A^{(k)}}^T\mathbf{r}^{(k+1)} = -\mathbf{A^{(k)}}^T\mathbf{r}^{(k)}. \tag{4.23}$$

If $\mathbf{r}^{(k+1)}$ were known, then the solution can be found easily. However, this is not the case, of course. Levenberg substituted $\lambda\Delta\mathbf{x}^k$ for $-\mathbf{A}^T\mathbf{r}^{(k+1)}$ in [35]. Therefore, (4.23) is transformed to

$$(\mathbf{A}^{(k)^T}\mathbf{A}^{(k)} + \lambda\mathbf{I}) \cdot \Delta\mathbf{x}^k = -\mathbf{A}^{(k)}\mathbf{r}^{(k)}, \tag{4.24}$$

where $\lambda \in \mathbb{R}$ is a scaling factor. It is evident that the method reduces to Newton's method when $\lambda = 0$.

Marquardt suggested that scaling the components of the gradient using the curvature ensures greater movement in directions in which the gradient has small magnitude [38]. Therefore, Marquardt replaced the identity matrix $\mathbf{I}$ in (4.24) with the diagonal matrix consisting of the diagonal elements of $\mathbf{A}^T\mathbf{A}$. Marquardt also changed the scaling value, λ, to $\lambda^{(k)}$, i.e., the scaling factor gets updated at each step. The value of $\lambda^{(k)}$ varies depending on the behavior of each iteration. To this end, at each step, a new parameter

$\nu^{(k)} \in [2, 10]$ is introduced. Then $\lambda^{(k)}$ is updated as follows.

$$\lambda^{(k+1)} = \begin{cases} \lambda^{(k)}/\nu^{(k)} & \text{if convergence is slow and stable;} \\ \nu^{(k)}\lambda^{(k)} & \text{if the iterations appear to be diverging.} \end{cases} \tag{4.25}$$

As a result, the Levenberg-Marquardt algorithm at the k-th iteration can be summarized as

$$(\mathbf{A^{(k)^T}A^{(k)}} + \lambda^{(k)}\text{diag}(\mathbf{A^{(k)^T}A^{(k)}})) \cdot \Delta\mathbf{x}^k = -\mathbf{A^{(k)^T}r}^{(k)}. \tag{4.26}$$

Later, Fletcher improved the algorithm by introducing a new and more effective method for updating λ at each step [4, 22] and [23, p. 100]. We refer to this method as LMF. For more details about the LMF method, refer to [4, 22, 23, 41, 44].

4.6.1 LMF for the polyenergetic problem

Recall our goal is to find the minimizer of

$$\min_{\boldsymbol{\mu}} G(\boldsymbol{\mu}), \tag{4.27}$$

where the objective function is

$$G(\boldsymbol{\mu}) = \|\mathbf{F}(\boldsymbol{\mu})\|_2^2, \tag{4.28}$$

and $\mathbf{F} = \{\mathbf{F}_i\}_{i=1}^{N_{\text{proj}}}$ is defined by

$$\mathbf{F}_i(\boldsymbol{\mu}) = \sum_{h=1}^{N_\varepsilon} \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h) - \mathbf{P}_i, \tag{4.29}$$

where $\boldsymbol{\mu}_h$ is a N_{grid}-vector representing the attenuation coefficient at the h-th energy level. In our implementation of the LMF, we use $\mathbf{F}(\boldsymbol{\mu})$ in (4.28) as the input. In (4.27), (4.28) and (4.29), $\boldsymbol{\mu}$ is of size $N_{\text{grid}} \times N_\varepsilon$, where N_{grid} is the number of pixels in the reconstruction

image and N_ε is the number of energy levels.

Recall also from (4.11) that the elements of the Jacobian matrix $\mathbf{A}$ are

$$\begin{aligned}\frac{\partial \mathbf{F}_i(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}} &= \frac{\partial}{\partial \boldsymbol{\mu}_{j,h}}\{\sum_{h^*=1}^{N_\varepsilon} \mathbf{w}_{h^*} \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_{h^*}) \exp(-\mathbf{R}_i^T \boldsymbol{\mu}_{h^*}) - \mathbf{P}_i\} \\ &= \sum_{h^*=1}^{N_\varepsilon} \mathbf{w}_{h^*} \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_{h^*}) \frac{\partial}{\partial \boldsymbol{\mu}_{j,h}}\{\exp(-\mathbf{R}_i^T \boldsymbol{\mu}_{h^*})\} \\ &= \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h)(-\mathbf{R}_{i,j})\{\exp(-\mathbf{R}_i^T \boldsymbol{\mu}_h)\}, \end{aligned} \tag{4.30}$$

where, as seen in (4.6), $\mathbf{R}$ is the Radon matrix of size $N_{\text{proj}} \times N_{\text{grid}}$. If we use (4.29) and (4.30) in (4.26), we get

$$(\mathbf{A}^{(\mathbf{k})\mathbf{T}} \mathbf{A}^{(\mathbf{k})} + \lambda^{(k)} \mathbf{B}) \cdot \Delta \boldsymbol{\mu}^{(k)} = -\mathbf{A}^{(\mathbf{k})\mathbf{T}} \mathbf{r}^{(k)}, \tag{4.31}$$

where $\mathbf{r}^{(k)} = \left[\mathbf{F}_1^{(k)}, \mathbf{F}_2^{(k)}, \ldots, \mathbf{F}_{N_{\text{proj}}}^{(k)}\right]$, and $\mathbf{F}_i(\boldsymbol{\mu})$ is defined in (4.29); $\mathbf{A}_{i,l}^{(k)} = \frac{\partial \mathbf{F}_i^{(k)}(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}}$, where $l = (h-1)N_{\text{proj}} + j$, $h = 1, \ldots, N_\varepsilon$ and $j = 1, \ldots, N_{\text{grid}}$; $\mathbf{B} = \text{diag}(\mathbf{A}^{(k)^T} \mathbf{A}^{(k)})$; and $\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \Delta \boldsymbol{\mu}^{(k)}$.

4.6.2 Shortcomings of the LMF approach

The LMF method described in Section 4.6 returns results with better image quality than the Newton-like method described in Section 4.5 and FBP (see Section 4.8). However, the main disadvantage of this method is that we have to store all the matrices in (4.31), which may require a very large amount of memory for the problems that arise in practice. In our preliminary test runs, we either experienced very long computation times or we ran out of memory. Later in Chapter 5, we discuss this deficiency in more detail and introduce some extensions to our method to overcome it.

4.7 An energy-dependent mathematical phantom

Based on the energy-dependent model introduced in Section 3.3.1, we developed an energy-dependent mathematical phantom for our experiments. We extended the MATLAB `phantom` function and called it `ephantom`. Our new MATLAB function generates an energy-dependent image of a head phantom that can be used to test the numerical accuracy of 2-D energy-dependent reconstruction algorithms. If we assume that we have N_ε different energy levels, `ephantom` returns N_ε grayscale intensity images. Each image represents the energy-dependent attenuation coefficient at that given energy level.

Each of these images consist of one large ellipse (representing the brain) containing several smaller ellipses (representing features in the brain). At each energy level, for any given pixel in the output image, the pixel's value is equal to the sum of the additive intensity values of all ellipses of which the pixel is a part. If a pixel is not part of any ellipse, its value is 0. Also note that, the additive intensity value for an ellipse can be positive or negative; if it is negative, the ellipse will be darker than the surrounding pixels.

Each ellipse is associated with an energy function $\mathbf{f}^n(\boldsymbol{\varepsilon})$, where $\boldsymbol{\varepsilon}$ is the energy and $n = 1, \ldots, N_{\text{ellipse}}$, where N_{ellipse} is the number of ellipses in our energy-dependent phantom. We have N_ε different energy levels, so

$$\mathbf{f}^n(\boldsymbol{\varepsilon}) = (\mathbf{f}^n(\boldsymbol{\varepsilon}_1), \ldots, \mathbf{f}^n(\boldsymbol{\varepsilon}_{N_\varepsilon})), \tag{4.32}$$

for $n = 1, \ldots, N_{\text{ellipse}}$. Therefore, for each energy level, the phantom at pixel $\mathbf{x}$ is computed by

$$\boldsymbol{\mu}_h(\mathbf{x}) = \sum_{n=1}^{N_{\text{ellipse}}} \mathbf{f}^n(\boldsymbol{\varepsilon}_h) C_n O_n(\mathbf{x}), \tag{4.33}$$

where C_n is a constant function associated with each ellipse, and

$$O_n(\mathbf{x}) = \begin{cases} 1 & \text{if pixel } \mathbf{x} \text{ is part of ellipse n,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.34}$$

As seen in Figure 4.3, as the energy increases, the attenuation coefficient decreases. Although, the materials shown in Figure 4.3 do not normally occur in the body, the dependence of the attenuation coefficient on energy is similar for the materials that do occur in the body. The sinogram data that is used as the input to test our reconstruction

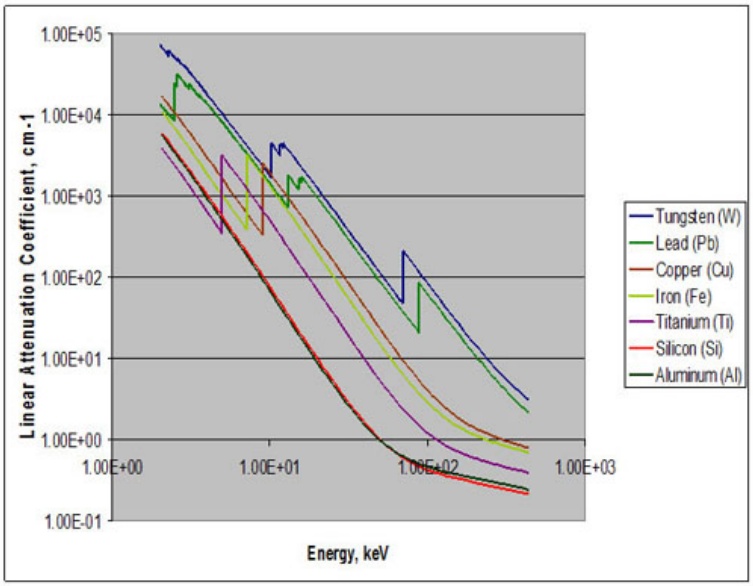


Figure 4.3: This image presents the relation between the value of the attenuation coefficient and the energy value for different materials.

algorithm is computed by the discretized polyenergetic physical model (4.8).

4.8 Numerical results

In our preliminary experiments, we used different MATLAB functions in the Optimization Toolbox as our solver, such as `fminunc`. This function finds the minimum of our unconstrained multivariable objective function. This algorithm is a subspace trust-region

method and is based on the interior-reflective Newton method [15, 14]. Each iteration involves the approximate solution of a large linear system using the preconditioned conjugate gradient (PCG) method.

We also experimented with the `lsqnonlin` function in the Optimization Toolbox. This function is useful in solving nonlinear least-squares problems. It uses either the trust-region-reflective optimization algorithm or the Levenberg-Marquardt method. In our case, since the reconstruction problems of interest to us are mostly underdetermined, we choose to use the Levenberg-Marquardt algorithm. The main problem is that, when we use these solvers for large data sets, we experience very long computation times or we run out of memory. However, for small data sets, using this algorithm, we obtain promising results, reported later in this section.

The two numerical experiments reported below are based on our energy-dependent mathematical phantom with

$$\mathbf{f}^n(\boldsymbol{\varepsilon}) = \frac{1}{1 + 0.5\boldsymbol{\varepsilon} \cdot n}, \tag{4.35}$$

where n is the index of the ellipse, as explained in Section 4.7. They are also based on a Gaussian distribution for $\mathbf{P}_0$, since, as seen in Figure 4.1, for low maximum energy values, we have a Gaussian-like distribution for $\mathbf{P}_0$. We use (4.2) with a Gaussian quadrature rule with $N_\varepsilon = 7$ nodes and weights to compute the sinogram data $\mathbf{P}_{t,\theta}$ with $N_\theta = 18$ and $N_t = 33$ (hence, $N_{\text{proj}} = N_\theta \times N_t = 594$). In the two numerical experiments reported below, we use $N_{\text{grid}} = 20 \times 20 = 400$.

Also, for each numerical experiment, we added Gaussian noise to the phantom, from which we make the sinogram data $\mathbf{P}_{t,\theta}$. To be more specific, here and for all the other numerical results in the thesis, we use the following MATLAB code to compute the Gaussian noise.

```
Gaussian_noise = mean(mu(:))*C_noise*var(mu(:))*randn(size(mu)),
```

where we use different constants C_{noise} to vary the amount of added noise.

In Figure 4.4, we display the sinogram data with $C_{\text{noise}} = 0.1$ used as input to the first of the two numerical experiments reported in this section to test our reconstruction algorithms. The sinogram for the second numerical experiment with $C_{\text{noise}} = 0.5$ is similar, so we have not displayed it.

In the rest of the figures in this section, we display the reconstructed images computed by our new nonlinear iterative algorithm using a Gaussian quadrature rule with $N_\varepsilon = 3$ nodes and weights. The two numerical experiments reported here are similar, except that, in the first, reported in Figures 4.5 and 4.6, we used $C_{\text{noise}} = 0.1$ and, in the second, reported in Figures 4.7 and 4.8, we used $C_{\text{noise}} = 0.5$.

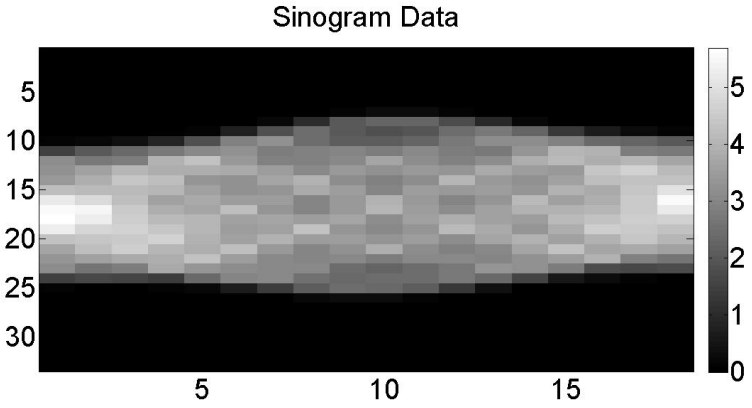


Figure 4.4: The sinogram data of the energy-dependent phantom computed by `ephantom.m`, with $N_{\text{grid}} = 20 \times 20$, $N_\varepsilon = 7$, $N_\theta = 18$, $N_t = 33$ and $C_{\text{noise}} = 0.1$. The sinogram for the second numerical experiment with $C_{\text{noise}} = 0.5$ is similar.

In Figures 4.5 and 4.7, the first row contains three slices of the energy-dependent Shepp-Logan phantom produced by our MATLAB function `ephantom.m` at the three energy levels used in these numerical experiments. The second row shows three energy-dependent slices of the reconstructed image computed by our new nonlinear algorithm. Finally, the third row shows the starting guess (the FBP solution) for the three energy levels.

In Figures 4.6 and 4.8, we merge the solutions for different energy levels using the discretized model (4.8). In these images, it is clear that the quality of the results are better

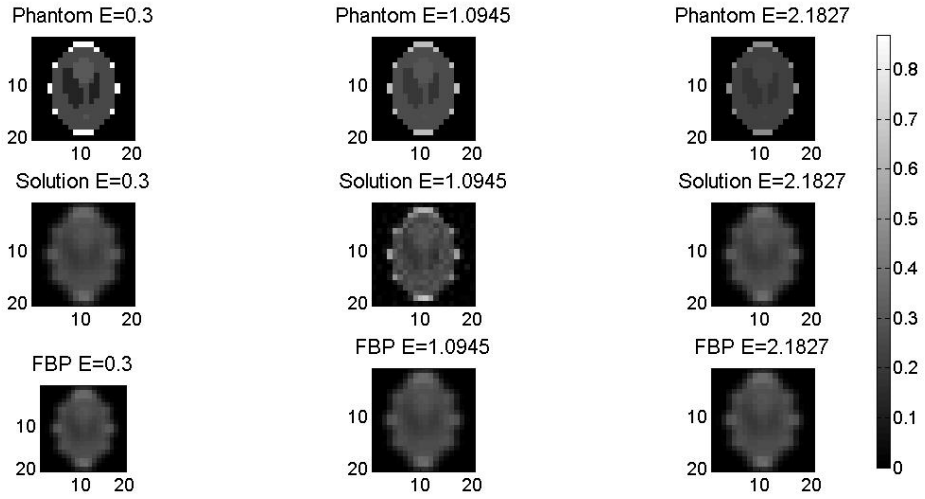


Figure 4.5: In this figure, we display the results with $N_{\text{grid}} = 20 \times 20$, $N_\varepsilon = 3$, $N_\theta = 18$, $N_t = 33$, and $C_{\text{noise}} = 0.1$. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new method. The third row shows the starting guess (the FBP solution) which is the same for each energy level.

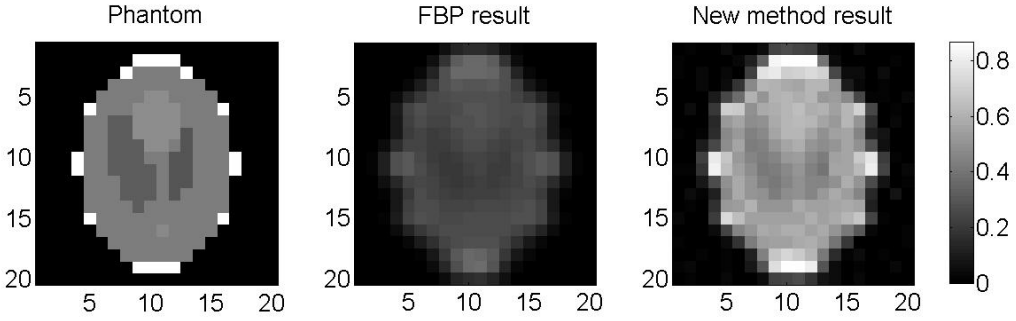


Figure 4.6: In this figure, we display the results with $N_{\text{grid}} = 20 \times 20$, $N_\varepsilon = 3$, $N_\theta = 18$, $N_t = 33$, and $C_{\text{noise}} = 0.1$. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8).

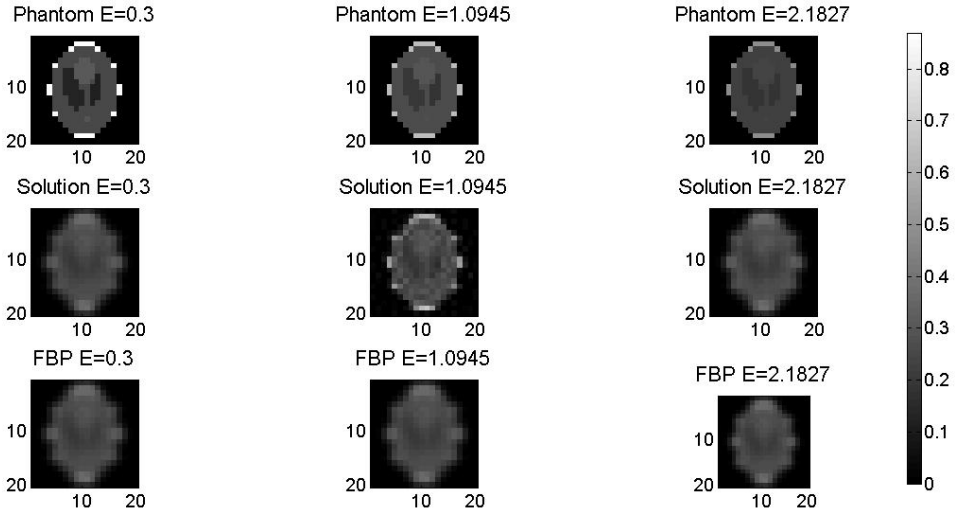


Figure 4.7: In this figure, we display the results with $N_{\text{grid}} = 20 \times 20$, $N_\varepsilon = 3$, $N_\theta = 18$, $N_t = 33$, and $C_{\text{noise}} = 0.5$. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new method. The third row shows the starting guess (the FBP solution) which is the same for each energy level.

for our algorithm than for FBP, which uses the Hamming filter and linear interpolation in the `iradon` function in MATLAB. However, there is no filtering or regularization involved in our reconstruction algorithm for these numerical results.

In these computations, 12 iterations of the Levenberg-Marquardt algorithm in the MATLAB code `lsqnonlin` were required for convergence to within the prescribed tolerance of 1e-10. The execution time for our method is approximately 90 seconds and for FBP is approximately 0.66 seconds.

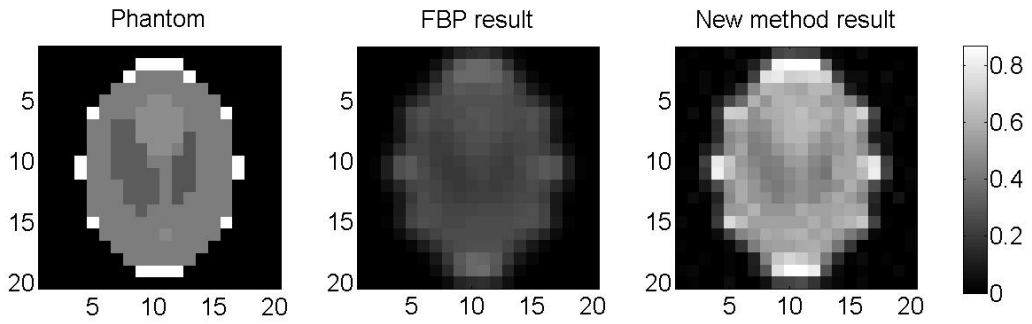


Figure 4.8: In this figure, we display the results with $N_{\text{grid}} = 20 \times 20$, $N_\varepsilon = 3$, $N_\theta = 18$, $N_t = 33$, and $C_{\text{noise}} = 0.5$. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8).

Chapter 5

Large Scale Polyenergetic Nonlinear Iterative Method

As a reminder, in Chapter 4, we present a nonlinear iterative reconstruction method based on the polyenergetic model in (4.1). We use a quadrature rule, such as a Gaussian quadrature rule or the composite Trapezoidal rule, and FBP and the Radon transform to discretize the energy-dependent continuous model with respect to different energy levels and different projections. To solve this problem, we study different nonlinear unconstrained optimization solvers, such as a Newton-like method and the Levenberg-Marquardt-Fletcher algorithm.

We find that the LMF method is the most effective among the methods we explored. However, in our implementation of this method, described in Section 4.6, we have to store all the matrices in (4.31). Hence, the memory requirements are large and the computation time is usually long. In fact, in some test problems, we run out of memory and the computation aborts. On the positive side, in Chapter 4, we report some promising results for our new method, compared to FBP solutions, on small non-clinical data sets.

Clinical data sets are usually very large. Therefore, the reconstruction method discussed in Section 4.6 may not be applicable. Hence, the goal of this chapter is to make

our algorithm more efficient for large scale problems. We explain in this chapter how this can be done.

5.1 Modified LMF algorithm for the polyenergetic problem

As discussed in Section 4.6, if we use (4.29) and (4.30) in (4.26), we get (4.31). To recap, the equation that we must solve at each LMF iteration is

$$(\mathbf{A}^{(k)^T}\mathbf{A}^{(k)} + \lambda^{(k)}\mathbf{B}^{(k)^T}) \cdot \Delta\boldsymbol{\mu}^k = -\mathbf{A}^{(k)^T}\mathbf{r}^{(k)}, \tag{5.1}$$

where $\mathbf{r}^{(k)} = \mathbf{r}(\boldsymbol{\mu}^{(k)}) = \left[\mathbf{F}_1(\boldsymbol{\mu}^{(k)}), \mathbf{F}_2(\boldsymbol{\mu}^{(k)}), \ldots, \mathbf{F}_{N_{\text{proj}}}(\boldsymbol{\mu}^{(k)})\right]$, $\mathbf{F}_i(\boldsymbol{\mu}^{(k)})$ is defined in (4.29), $\mathbf{A}^{(k)}_{i,l} = \frac{\partial \mathbf{F}_i(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_{j,h}}$, for $l = (h-1)N_{\text{proj}}+j$, $\mathbf{B}^{(k)} = \text{diag}\left(\mathbf{A}^{(k)^T}\mathbf{A}^{(k)}\right)$ and $\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \Delta\boldsymbol{\mu}^{(k)}$. To reduce the clutter of notation, we drop the superscript k from now on.

One of the main disadvantages of our method as presented in Chapter 4 is that we have to store all the large and not-so-sparse matrices. In the next sections, we introduce a variant of the method that overcomes this deficiency. One difficulty that arises in our new variant is computing the diagonal matrix $\mathbf{B}$. We modify the method and use instead a block diagonal matrix for $\mathbf{B}$, as is explained in Section 5.3.1.

5.2 The Radon matrix in the Jacobian matrix

As explained in Section 4.6.1, the elements of the Jacobian matrix $\mathbf{A} = [\mathbf{A}_{i,m}]$ are

$$\mathbf{A}_{i,l} = \mathbf{w}_h \cdot \mathbf{P}_0(\boldsymbol{\varepsilon}_h)(-\mathbf{R}_{i,j})\{\exp(-\mathbf{R}_i^T\boldsymbol{\mu}_h)\}, \tag{5.2}$$

where $l = (h-1)N_{\text{proj}} + j$ and $\mathbf{R}_i^T$ is the i-th row of $\mathbf{R}$, for $i = 1, \ldots, N_{\text{proj}}$. Recall that we use $\mathcal{R}$ for the Radon transform and $\mathbf{R}$ for the associated Radon matrix (see

Equation (4.7)). From (5.2), we see that $\mathbf{A}$ has the form

$$\mathbf{A} = [\mathbf{C}^{(1)}\mathbf{R}, \mathbf{C}^{(2)}\mathbf{R}, \ldots, \mathbf{C}^{(N_\varepsilon)}\mathbf{R}], \tag{5.3}$$

where $\mathbf{C}^{(h)}$, $h = 1, \ldots, N_\varepsilon$, is the $N_{\text{proj}} \times N_{\text{proj}}$ diagonal matrix defined by

$$\mathbf{C}^{(h)} = \text{diag}([-\mathbf{w}_h \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_1 \boldsymbol{\mu}_h), \ldots, -\mathbf{w}_h \mathbf{P}_0(\boldsymbol{\varepsilon}_h) \exp(-\mathbf{R}_{N_{\text{proj}}} \boldsymbol{\mu}_h)]), \tag{5.4}$$

and $\mathbf{R}$ is the matrix associated with the Radon transform (see Section 2.2.1 and 4.2.2). Hence, when computing the Jacobian matrix $\mathbf{A}$, we can take advantage of the structure of the matrix $\mathbf{R}$, which is already known. Moreover, the structure of $\mathbf{R}$ is repeated N_ε (the number of energy levels) times, in the Jacobian. This is illustrated in Figure 5.1.

5.3 Using back-projection in LMF

In Section 4.6, we review the LMF algorithm and in (4.26) we specify the system of linear equations that must be solved at each iteration of the algorithm. In (5.3), we show that we can write the Jacobian matrix in terms of the Radon matrix. Therefore, we can write $\mathbf{A}^T\mathbf{A}$ in terms of the Radon matrix as well.

$$\begin{aligned} \mathbf{A}^T\mathbf{A} &= [\mathbf{C}^{(1)}\mathbf{R}, \ldots, \mathbf{C}^{(N_\varepsilon)}\mathbf{R}]^T \cdot [\mathbf{C}^{(1)}\mathbf{R}, \ldots, \mathbf{C}^{(N_\varepsilon)}\mathbf{R}] \\ &= \begin{pmatrix} \mathbf{R}^T\mathbf{C}^{(1)^T} \\ \mathbf{R}^T\mathbf{C}^{(2)^T} \\ \vdots \\ \mathbf{R}^T\mathbf{C}^{(N_\varepsilon)^T} \end{pmatrix} \cdot [\mathbf{C}^{(1)}\mathbf{R}, \ldots, \mathbf{C}^{(N_\varepsilon)}\mathbf{R}]. \end{aligned} \tag{5.5}$$

In these equations we use $\mathbf{R}^T$. Since $\mathbf{R}$ is a real-valued matrix, $\mathbf{R}^T = \mathbf{R}^*$ and can be approximated by $\mathcal{R}^*$. In Section 5.3.2, we explain how to relate the matrix $\mathbf{R}^*$ (or, more specifically, the operator $\mathcal{R}^*$) to back-projection.

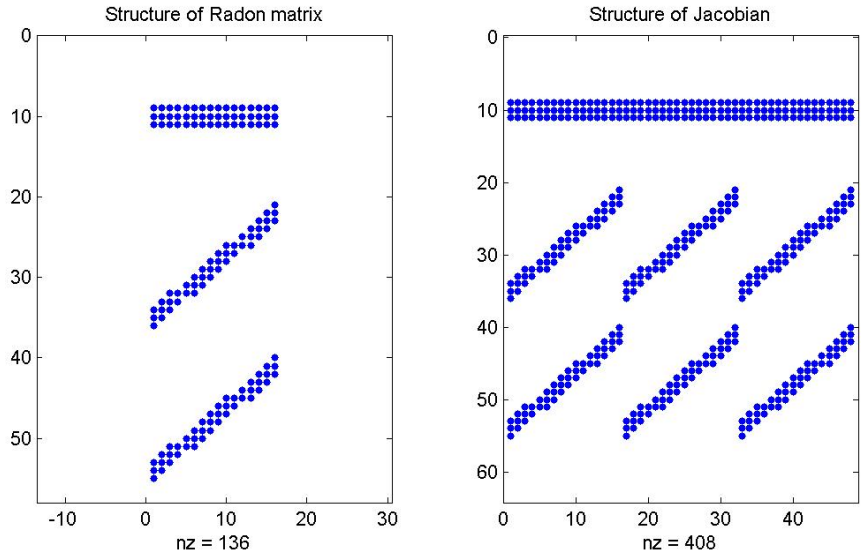


Figure 5.1: On the left, we display the sparsity structure of the Radon matrix, for a 4 by 4 image ($N_{\text{grid}} = 16$), $N_\theta = 3$ and $N_t = 19$ which makes $N_{\text{proj}} = 57$. On the right, we display the sparsity structure of the Jacobian with the same number of pixels and angles for 3 different energy levels ($N_\varepsilon = 3$). Note that the structure of the Radon matrix is repeated 3 times in the structure of the Jacobian matrix. We choose small values for N_{grid}, N_{proj} and N_ε to make the images clear.

5.3.1 Block diagonal matrix

As mentioned above, computing the diagonal of $\mathbf{A}^T\mathbf{A}$ can be computationally expensive. We modify the LMF method by using a block diagonal matrix for $\mathbf{B}$ in (4.31). To this end, note that $\mathbf{A}^T\mathbf{A}$ is of size $N_{\text{grid}}N_\varepsilon \times N_{\text{grid}}N_\varepsilon$. This matrix has $N_\varepsilon \times N_\varepsilon$ blocks each of size $N_{\text{grid}} \times N_{\text{grid}}$. The (i, j)-th block is

$$\mathbf{R}^T\mathbf{C}^{(i)}\mathbf{C}^{(j)}\mathbf{R}. \tag{5.6}$$

We take $\mathbf{B}$ in our modified LMF method to be

$$\mathbf{B} = \begin{pmatrix} \mathbf{R}^T\mathbf{C}^{(1)}\mathbf{C}^{(1)}\mathbf{R} & & & \\ & \mathbf{R}^T\mathbf{C}^{(2)}\mathbf{C}^{(2)}\mathbf{R} & & \\ & & \ddots & \\ & & & \mathbf{R}^T\mathbf{C}^{(N_\varepsilon)}\mathbf{C}^{(N_\varepsilon)}\mathbf{R} \end{pmatrix}. \tag{5.7}$$

Computing and storing the $\mathbf{C}^{(i)}$s is cheap, since they are diagonal. In the next sections we show that we do not need to compute or store $\mathbf{R}$ and $\mathbf{R}^*$. Instead, we can take advantage of our knowledge about the Radon transform and back-projection (see Sections 5.3.2 and 5.4.1) to develop what we call the matrix-free LMF method. The block-diagonal form of $\mathbf{B}$ shown in (5.7) is much easier to work with in this matrix-free version of our algorithm than $\mathbf{B} = \operatorname{diag}\left(\mathbf{A}^T\mathbf{A}\right)$, which occurs in the standard LMF algorithm.

5.3.2 Back-projection and the adjoint of the Radon transform

In [19, p. 197], Epstein shows the well-known result that back-projection equals $(4\pi)^{-1}$ times the formal adjoint of the Radon transform. We'll use this relationship between back-projection and the adjoint of the Radon transform to derive our matrix-free LMF method in Section 5.4.1. For completeness, we briefly review below Epstein's derivation of the relationship between back-projection and the adjoint of the Radon transform.

To this end, consider the same variables and notations used in Chapter 2. Assume $\mathbf{b}$ is a function of bounded support on $\mathbb{R} \times S^1$. The inner product of $\mathcal{R}\mu$ and $\mathbf{b}$ is

$$\begin{aligned} \langle \mathcal{R}\mu, \mathbf{b} \rangle &= \int_0^{2\pi} \int_{-\infty}^{+\infty} \mathcal{R}\mu(t,\theta)\mathbf{b}(t,\theta)\, dt\, d\theta \\ &= \int_0^{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu(t\boldsymbol{\omega}(\theta) + s\widehat{\boldsymbol{\omega}}(\theta))\mathbf{b}(t,\theta)\, ds\, dt\, d\theta. \end{aligned} \tag{5.8}$$

If we let $\mathbf{x} = t\boldsymbol{\omega}(\theta) + s\widehat{\boldsymbol{\omega}}(\theta)$, we get $t = \langle \mathbf{x}, \boldsymbol{\omega}(\theta) \rangle$. If we switch the order of the integrals, we can rewrite (5.8) as

$$\begin{aligned} \langle \mathcal{R}\mu, \mathbf{b} \rangle &= \int_{\mathbb{R}^2} \int_0^{2\pi} \mu(\mathbf{x}) \mathbf{b}(\langle \mathbf{x}, \boldsymbol{\omega}(\theta) \rangle, \theta) \, d\theta \, d\mathbf{x} \\ &= \langle \mu, \mathcal{R}^* \mathbf{b} \rangle. \end{aligned} \tag{5.9}$$

Equation (5.9) shows that back-projection is $(4\pi)^{-1}$ times the formal adjoint of the Radon transform. As noted above, we use this relationship in Section 5.4.1 to develop a matrix-free version of the LMF-based iterative polyenergetic reconstruction method.

5.4 Storage reduction and fast computation

Generally, two of the main disadvantages of iterative reconstruction methods, including the LMF described in Chapter 4, are high computational and storage costs. This makes such methods inefficient (or not even applicable) when applied to the large problems that arise in practice.

A few approaches to ameliorate the deleterious affects of large memory requirements are discussed briefly below.

We can divide a large data set into blocks and process one block at a time. In our case, we can process the data associated with each energy level separately. However, this approach does not help us much. We can also break any functions that are used into nested functions that require fewer arguments. This method allows us to reduce the memory requirements to some extent. However, it does not help to reduce our computation time. We may also apply other methods, such as using sparse arrays, when possible; making more efficient use of memory, using preallocation of a block of memory, large enough to hold the matrix at its final size before processing it; making sure to clear old variables from memory; and allocating large matrices first. These approaches help us

improve the performance of our method to some small extent. However, the improvement is not sufficient.

In addition to the standard approaches listed above, we develop a matrix-free iterative reconstruction method, based on the LMF algorithm, in which we do not need to form and save any matrices. This method helps us improve the performance of our reconstruction algorithm. In Section 5.4.1, we explain this method in detail.

5.4.1 Matrix-free computation

Once again, consider the linear system (4.31) that must be solved at each iteration of the LMF method. As mentioned before, we drop the superscript k to reduce the clutter of notation. Therefore, the equation at the k-th iteration of the LMF method can be written as

$$(\mathbf{A^T A} + \lambda \mathbf{B}) \cdot \Delta \boldsymbol{\mu} = -\mathbf{A^T r}. \tag{5.10}$$

We can use conjugate gradient (CG) method to solve (5.10). Computing matrices $\mathbf{A}$ and $\mathbf{A^T A}$ can be very time-consuming and storing them requires a lot of memory. Hence, we develop a matrix-free method, in which there is no need to construct and store the matrices in question. To implement CG with this approach, we compute $\mathbf{b} = -\mathbf{A}^T\mathbf{r}$ by using the transpose of (5.3). That is,

$$\mathbf{A}^T\mathbf{r} = \begin{pmatrix} \mathbf{R}^T\mathbf{C}^{(1)} \\ \mathbf{R}^T\mathbf{C}^{(2)} \\ \vdots \\ \mathbf{R}^T\mathbf{C}^{(N_\varepsilon)} \end{pmatrix} \cdot \mathbf{r} = \begin{pmatrix} \mathbf{R}^T\mathbf{C}^{(1)} \cdot \mathbf{r} \\ \mathbf{R}^T\mathbf{C}^{(2)} \cdot \mathbf{r} \\ \vdots \\ \mathbf{R}^T\mathbf{C}^{(N_\varepsilon)\cdot \mathbf{r}} \end{pmatrix} = \begin{pmatrix} \mathbf{R}^*\mathbf{C}^{(1)} \cdot \mathbf{r} \\ \mathbf{R}^*\mathbf{C}^{(2)} \cdot \mathbf{r} \\ \vdots \\ \mathbf{R}^*\mathbf{C}^{(N_\varepsilon)\cdot \mathbf{r}} \end{pmatrix}, \tag{5.11}$$

where we use $\mathbf{R}^T = \mathbf{R}^*$. As discussed in Section 5.3.2, $\mathbf{R}^*$ can be approximated by back-projection (BP). BP can be implemented very efficiently, as explained in Section 2.4, and

hence is much less computationally expensive than an explicit matrix multiply.

To solve (5.10) by CG, we need to be able to compute $\mathbf{y} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{B}) \cdot \mathbf{x}$ for any given vector $\mathbf{x}$. We partition this computation as

$$\mathbf{y} = (\mathbf{A^T A} + \lambda \mathbf{B}) \cdot \mathbf{x} = \underbrace{\mathbf{A^T} \underbrace{\mathbf{Ax}}_{\mathbf{w}}}_{\mathbf{v}} + \lambda \underbrace{\mathbf{Bx}}_{\mathbf{u}} \tag{5.12}$$

With this partitioning, we can do the computations in stages below without explicitly forming the matrices as follows.

1. Compute $\mathbf{w} = \mathbf{Ax}$, where $\mathbf{A}$ is defined by (5.3). More specifically,

$$\begin{aligned} \mathbf{Ax} &= [\mathbf{C}^{(1)}\mathbf{R}, \mathbf{C}^{(2)}\mathbf{R}, \ldots, \mathbf{C}^{(N_\varepsilon)}\mathbf{R}]_{N_{\text{proj}} \times N_{\text{grid}} N_\varepsilon} \cdot [\mathbf{x}_1^T, \ldots, \mathbf{x}_{N_\varepsilon}^T]^T_{N_{\text{grid}} N_\varepsilon \times 1} && (5.13) \\ &= \sum_{h=1}^{N_\varepsilon} \mathbf{C}^{(h)}\mathbf{R}\mathbf{x}_h. && (5.14) \end{aligned}$$

   In this equation $\mathbf{R}\mathbf{x}_h \approx \mathcal{R}\mathbf{x}_h$, the Radon transform defined in (2.3). Hence, the only matrices that need to be computed are the $\mathbf{C}^{(h)}$s, defined in (5.4). These matrices are diagonal; therefore, computing and storing them is not expensive.

2. Compute $\mathbf{v} = \mathbf{A}^T\mathbf{w}$ using the same approach used to compute $\mathbf{A}^T\mathbf{r}$ (see (5.11) and the following explanation).

3. Compute $\mathbf{u} = \mathbf{Bx}$, where $\mathbf{B}$ is a block diagonal matrix with the ith diagonal block being $\mathbf{R}^T\mathbf{C}^{(i)}\mathbf{C}^{(i)}\mathbf{R}$. Thus, this computation can be carried out in stages similar to those described in points 1 and 2 above, exploiting the relationship between $\mathbf{R}$ and the Radon transform and between $\mathbf{R}^T$ and BP.

4. Finally, we use the results of the previous steps and compute $\mathbf{y} = \mathbf{v} + \lambda\mathbf{u}$, which is equal to the right hand side of (5.12).

This version of our method requires far less storage than our earlier version and is also faster. Therefore, it allows us to reconstruct large images at different energy levels. We discuss the numerical results in Section 5.6.

5.5 Regularization and image restoration

In our CT image reconstruction problem, like other inverse problems, we are trying to estimate some quantities (the attenuation coefficient for CT image reconstruction) using some indirect measurements of these quantities (the sinogram data for CT image reconstruction). Inverse problems are often ill-conditioned. For CT image reconstruction, noise in the sinogram data may give rise to significant errors in the estimated attenuation coefficient. To overcome this ill-conditioning, *regularization* is frequently used. In image processing, this process may be interpreted as *filtering* or *image restoration*. The main goal of image restoration is noise removal, e.g., sensor noise, motion blur, etc.

To regularize our problem, we extend our objective function (4.28) to

$$\widehat{G}(\boldsymbol{\mu}) = G(\boldsymbol{\mu}) + \gamma\phi(\boldsymbol{\mu}), \tag{5.15}$$

where γ is called the regularization parameter and $\phi(\mu)$ is called the regularization function. The parameter γ controls the trade-off between fidelity in solving the original problem and smoothness, for small and large γ, respectively. The regularization function is often taken to be

$$\phi(\boldsymbol{\mu}) = \|\mathbf{D}\boldsymbol{\mu}\|, \tag{5.16}$$

or

$$\phi(\boldsymbol{\mu}) = \|\mathbf{D}\boldsymbol{\mu}\|^2, \tag{5.17}$$

where $\|\cdot\|$ is some norm, which does not have to be the same as the norm used in the objective function $G(\boldsymbol{\mu})$. Here $\mathbf{D}$ is called the *stabilizing operator*. $\mathbf{D}$ is normally a

very sparse matrix and the matrix-vector multiplication can often be implemented very efficiently using a *filter kernel*. For our problem, we choose to use a discrete differential operator (5.18). For more details about different stabilizing operators and filter kernels, refer to [11, 12, 36, 46].

$$\mathbf{D} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix}_{(N_{\text{grid}}N_\varepsilon - 1)\times(N_{\text{grid}}N_\varepsilon)} \tag{5.18}$$

5.5.1 Tikhonov regularization

In image processing, *Tikhonov* regularization is often used for restoring images contaminated by noise.

Consider the objective function

$$\widehat{G}(\boldsymbol{\mu}) = \|\mathbf{F}(\boldsymbol{\mu})\|_2^2 + \gamma\|\mathbf{D}\boldsymbol{\mu}\|_2^2, \tag{5.19}$$

where $\mathbf{D}$ is defined in (5.18). As seen before, we define our problem as a nonlinear least-squares fitting problem. Recall that $\mathbf{F}(\boldsymbol{\mu})$ defined in (4.9) is of size N_{proj}. Also, the attenuation coefficient $\boldsymbol{\mu}$ is of size $N_{\text{grid}} \times N_\varepsilon$ or if we write it as a vector, of size $N_{\text{grid}}N_\varepsilon \times 1$. Let

$$\mathbf{D}\boldsymbol{\mu} = [\ddot{\boldsymbol{\mu}}_1, \ddot{\boldsymbol{\mu}}_2, \dots, \ddot{\boldsymbol{\mu}}_{N_{\text{grid}}N_\varepsilon - 1}]^T. \tag{5.20}$$

As mentioned in Section 4.6.1, we use $\mathbf{F}(\boldsymbol{\mu})$ in (4.28) as the input to our implementation. To use the same implementation for our regularized problem, we replace $\mathbf{F}(\boldsymbol{\mu})$ by

$$[\mathbf{F}_1(\boldsymbol{\mu}), \mathbf{F}_2(\boldsymbol{\mu}), \dots, \mathbf{F}_{N_{\text{proj}}}(\boldsymbol{\mu}), \gamma^{1/2}\ddot{\boldsymbol{\mu}}_1, \gamma^{1/2}\ddot{\boldsymbol{\mu}}_2, \dots, \gamma^{1/2}\ddot{\boldsymbol{\mu}}_{N_{\text{grid}}N_\varepsilon - 1}]^T. \tag{5.21}$$

Consequently the Jacobian matrix defined in (4.30) and (5.3) can be extended to

$$\widetilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \gamma^{1/2}\,\mathbf{D} \end{pmatrix}, \tag{5.22}$$

where $\mathbf{D}$ is defined in (5.18). Hence, we can formulate (5.19) as a standard nonlinear least squares (NLS) problem. This regularization transforms an underdetermined NLS problem into an overdetermined one. Now we can follow the approach described in Section 4.6 to solve the new regularized NLS problem (5.21).

5.5.2 Regularization in the 1-norm

In this method, we define the regularization function as

$$\Phi(\boldsymbol{\mu}) = \|\mathbf{D}\boldsymbol{\mu}\|_1, \tag{5.23}$$

where $\mathbf{D}$ is defined in (5.18). That is, we can regularize our objective function using $\Phi(\boldsymbol{\mu})$ and rewrite the problem as

$$\widehat{G}(\boldsymbol{\mu}) = \|\mathbf{F}(\boldsymbol{\mu})\|_2^2 + \gamma\Phi(\boldsymbol{\mu}) \equiv \|\mathbf{F}(\boldsymbol{\mu})\|_2^2 + \gamma\|\mathbf{D}\boldsymbol{\mu}\|_1. \tag{5.24}$$

In the Levenberg-Marquardt-Fletcher method, we optimize a nonlinear least-squares fitting problem in the 2-norm. As mentioned in Sections 4.6.1 and 5.5.1, we use $\mathbf{F}(\boldsymbol{\mu})$ in (4.28) as the input to our implementation. If we want to use the same implementation, we have to use some tricks for regularization in the 1-norm to approximate $\|\mathbf{D}\boldsymbol{\mu}\|_1$ in (5.24) by a 2-norm. To this end, we use the iterative reweighted norm algorithm. For more details about regularization in the 1-norm, see [26], [52], and [56].

Define the regularization vector and objective function as we do in (5.20) and (5.21). However, since this time we want to optimize the regularization function in the 1-norm,

we need to include weights ν_j for $j = 1, \ldots, N_{\text{grid}}N_\varepsilon$. We use an iterative method for computing the weights. In this method, we use the approximate value of $\boldsymbol{\mu}$ in the previous iteration, i.e., $\boldsymbol{\mu}^{(k-1)}$, to compute the new weights at iteration k. That is, at iteration k, we replace $\mathbf{F}(\boldsymbol{\mu})$ in (4.10) by

$$[\mathbf{F}_1(\boldsymbol{\mu}^{(k)}), \mathbf{F}_2(\boldsymbol{\mu}^{(k)}), \ldots, \mathbf{F}_{N_{\text{proj}}}(\boldsymbol{\mu}^{(k)}), \gamma^{1/2}\nu_1\boldsymbol{\mu}_1^{(k)}, \gamma^{1/2}\nu_2\boldsymbol{\mu}_2^{(k)}, \ldots, \gamma^{1/2}\nu_{N_{\text{grid}}N_\varepsilon-1}\boldsymbol{\mu}_{N_{\text{grid}}N_\varepsilon-1}{}^{(k)}]^T, \tag{5.25}$$

where $\nu_j = (\left|\ddot{\boldsymbol{\mu}}_j^{(k-1)}\right|)^{-1/2}$. Consequently, we modify the Jacobian matrix as follows. Define

$$\widetilde{\mathbf{D}} = \begin{pmatrix} \nu_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \nu_{N_{\text{grid}}N_\varepsilon-1} \end{pmatrix} \cdot \mathbf{D}, \tag{5.26}$$

where $\mathbf{D}$ is defined in (5.18). Now the Jacobian is extended to

$$\widetilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \gamma^{1/2}\widetilde{\mathbf{D}} \end{pmatrix}. \tag{5.27}$$

5.6 Numerical results

In Section 4.8, we present numerical results for a 20 by 20 energy-dependent phantom ($N_{\text{grid}} = 20 \times 20 = 400$) at 3 different energy levels ($N_\varepsilon = 3$), using $N_\theta = 18$ and $N_t = 33$ ($N_{\text{proj}} = N_\theta \times N_t = 594$). However, due to the small size of the projection data, this is not a realistic example. Unfortunately, using the methods described in Chapter 4, we are not able to handle larger, more realistic, reconstruction problems. In this section, we show that by applying the methods developed in this chapter, we can significantly increase the image resolution and projection angles and get satisfactory results.

We present four different experiments to assess the effectiveness of our algorithm. In each numerical example, we use $N_\varepsilon = 7$ energy levels, $N_\theta = 180$ projection angles,

$N_t = 287$ affine parameters (note that $N_{\text{proj}} = N_\theta \times N_t = 51,660$), and $N_{\text{grid}} = 200 \times 200 = 40,000$ grid points (note that number of unknowns is $N_{\text{grid}} \times N_\varepsilon = 280,000$). In each example, we either use no regularization, Tikhonov regularization or regularization in the 1-norm. In these experiments, we use the same energy-dependent phantom described in Section 4.7. We also add some Gaussian noise to the data as explained in Section 4.8. These example are summarized in the following table.

	Noise constant	Regularization method
Example 1	0	None
Example 2	0.5	None
Example 3	0.5	Tikhonov ($\gamma = 10^{-3}$)
Example 4	0.5	1-norm ($\gamma = 10^{-3}$)

In these examples, for $\mathbf{P}_0$ we use the non-smooth double-peaked 120keV distribution shown in Figure 4.1. We use with $\varepsilon_{\min} = 10$ and $\varepsilon_{\max} = 120$. We apply the composite Trapezoidal rule on a nonuniform grid with $N_\varepsilon = 7$ nodes to discretize the outer integral in (4.1) with respect to different energy levels. See Figure 5.2 for the placement of the nodes. Note that, if we had used a smooth Gaussian distribution (similar to the 60keV distribution in Figure 4.1), we could have used a Gaussian quadrature rule instead.

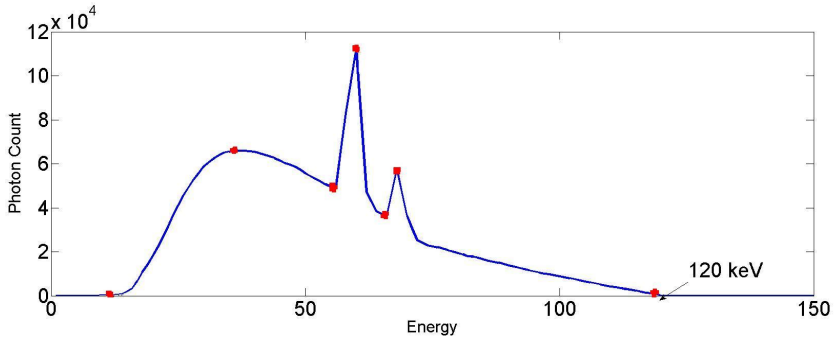


Figure 5.2: In this figure, we display the 7 nonuniform nodes (red dots) for the composite Trapezoidal rule used in Examples 1, 2, 3 and 4.

To make the mathematical phantom, we use the same energy function

$$\mathbf{f}^n(\boldsymbol{\varepsilon}) = \frac{1}{1 + 0.5\boldsymbol{\varepsilon} \cdot n}, \tag{5.28}$$

that we used for our examples reported in Section 4.8, where n in (5.28) is the index of the ellipse, as explained in Section 4.7. We compute the sinogram data by the discretized polyenergetic physical model (4.8) using $N_\varepsilon = 11$ different energy levels (larger than the N_ε we use in the quadrature rule for the reconstruction stage). Therefore, we do not take advantage of any additional information about the phantom or sinogram data. The sinogram data use in Example 1 is displayed in Figure 5.3. The sinogram data used in the other examples looks similar.

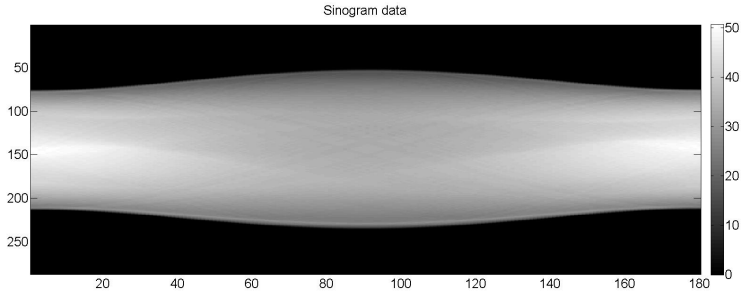


Figure 5.3: The sinogram data of the energy-dependent phantom computed by `ephantom.m`, with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 11$, $N_\theta = 180$ and $N_t = 287$. This sinogram data is used in Example 1. Similar sinogram data with some added noise is used in Examples 2, 3 and 4.

For each example, we plot the results for our new matrix-free LMF-based polyenergetic reconstruction algorithm at different energy levels. Then we compare these results with the FBP results and the phantom. To find the starting guess (FBP solution), we use the MATLAB function `iradon` with linear interpolation and the Hamming filter.

We also plot $\widehat{G}(\boldsymbol{\mu}^{(k)})$ versus the number of iterations. In each example, we see that, after several iterations of our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm, the value of the objective function at the final iterate is much smaller

than for the FBP solution that we use as the starting guess for our new algorithm. We also plot $\lambda^{(k)}$ versus the number of iterations.

For each example, we also plot the image profile of the phantom, the FBP result and the numerical solution using our new method. The profile of an image is the set of intensity values taken from regularly spaced points along a line segment or multiline path in an image. We use one path which contains many details and plot that.

Finally, for each example, we report the execution time for both FBP and our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm; the tolerance used for the stopping criterion for the LMF method; the tolerance used for CG; the total number of iterations of the LMF method; and the total number of CG iterations. Note that the execution time for the `radon` function on a 200×200 grid and $N_\theta = 180$ is approximately 0.89 seconds. The execution time for the `iradon` function for a 180×287 sinogram using linear interpolation and no filtering (which we use for BP to approximate $\mathbf{R}^T$) is approximately 1.43 seconds. The execution time for the `iradon` function using the Hamming filter and linear interpolation (which we use for FBP) is approximately 1.75 seconds.

We ran these experiments in Matlab version R2008b on a laptop with a 2.00GHz dual core Intel CPU with 2GB memory that runs Windows XP.

5.6.1 Example 1

In Example 1, we add no noised to the input data and apply no regularization. The numerical results for Example 1 are presented in Figures 5.4, 5.5, 5.6 and 5.7.

In this experiment, the execution time for our new method is approximately 2437.21 seconds. We use the tolerance 10^{-3} and the maximum number of iteration of 10 for the stopping criteria for the LMF method. The tolerance used for CG at each LMF iteration is 10^{-1}. This is an unusually relaxed tolerance, but, in numerical experiments, we found that it produced as accurate image reconstructions at less computational cost than the

more stringent tolerances that we tried. In this example, we use 10 LMF iterations and the total number of CG iterations is 20.

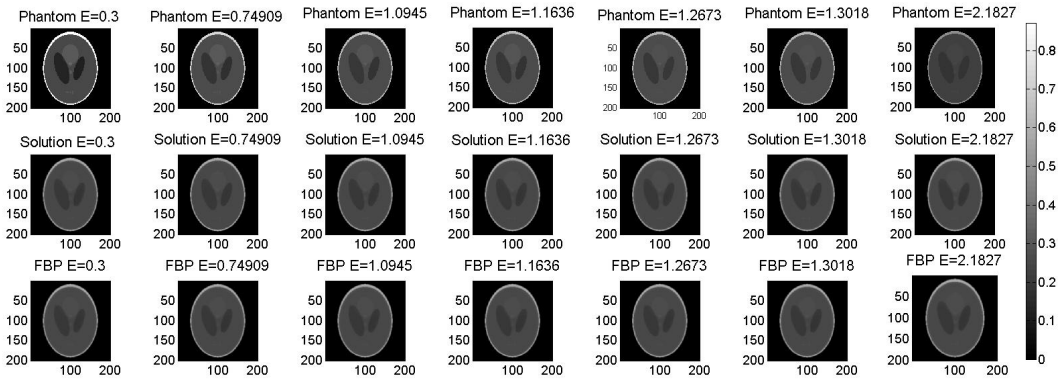


Figure 5.4: In this figure, we display the results for Example 1 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, adding no noise and applying no regularization. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new matrix-free polyenergetic nonlinear algorithm. The third row shows the starting guess (FBP solution) images which are the same for each energy level. It is clear that the results for our new algorithm are significantly better than the results for FBP.

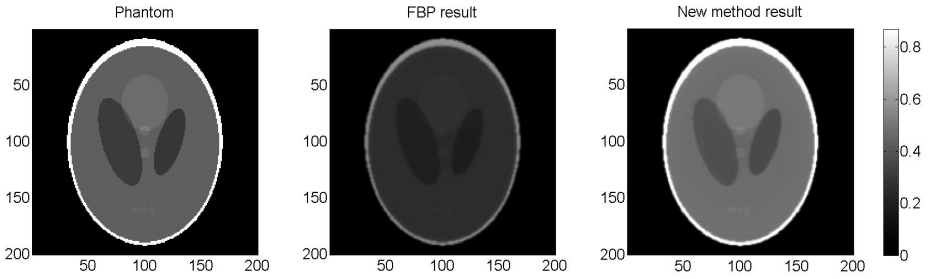
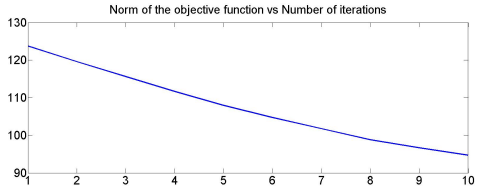
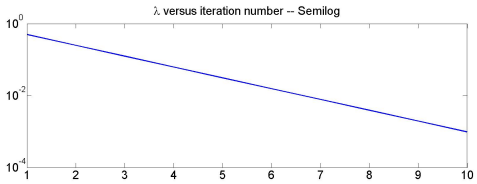


Figure 5.5: In this figure, we display the results for Example 1 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, adding no noise and applying no regularization. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8). It is clear that the results for our new algorithm are significantly better than the results for FBP.



(a) $\widehat{G}(\boldsymbol{\mu}^{(k)})$ versus the number of iterations. We see that, after 10 iterations of our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm, $\widehat{G}(\boldsymbol{\mu}^{(k)})$ is smaller than for the FBP solution that we use as the starting guess for our new algorithm (compare the first and last points of the curve).



(b) $\lambda^{(k)}$ versus the number of iterations (see Equation (4.31)) for Example 1.

Figure 5.6: In this figure, we display the results for Example 1 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, adding no noise and applying no regularization.

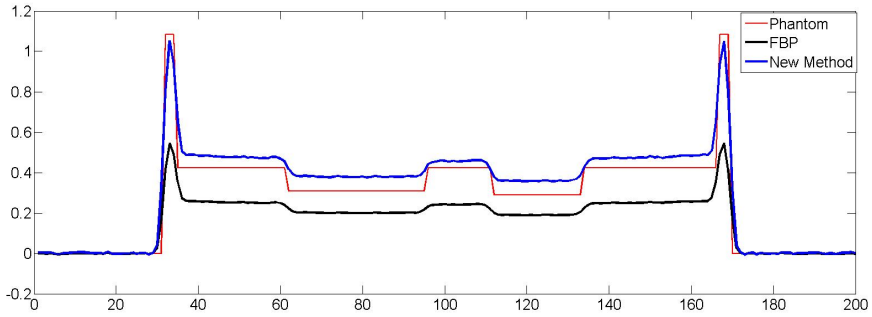


Figure 5.7: In this figure, we display the profile of the phantom, the FBP result and the numerical solution computed by our new method for Example 1 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, adding no noise and applying no regularization. It is clear that the profile for our new method is much closer to the profile for the phantom than is the profile for FBP.

5.6.2 Example 2

Example 2 is similar to Example 1 above, except that we add noise to the input data. As explained in Section 4.8, for each numerical experiment, we add Gaussian noise to the phantom, from which we make the sinogram data $\mathbf{P}_{t,\theta}$. To be more specific, here and for all the other numerical results, we use the following MATLAB code to compute the Gaussian noise.

```
Gaussian_noise = mean(mu(:))*C_noise*var(mu(:))*randn(size(mu)),
```

where we use different constants C_{noise} to vary the amount of added noise.

In this example, we use $C_{\text{noise}} = 0.5$. Like Example 1, we apply no regularization. We present the numerical results for this example in Figures 5.8, 5.9, 5.10 and 5.11.

In this experiment, the execution time for our new method is approximately 2670.60 seconds. We use the tolerance 10^{-3} and the maximum number of iteration of 10 for the stopping criteria for the LMF method. The tolerance used for CG at each LMF iteration is 10^{-1}. In this example, we use 10 LMF iterations and the total number of CG iterations is 22.

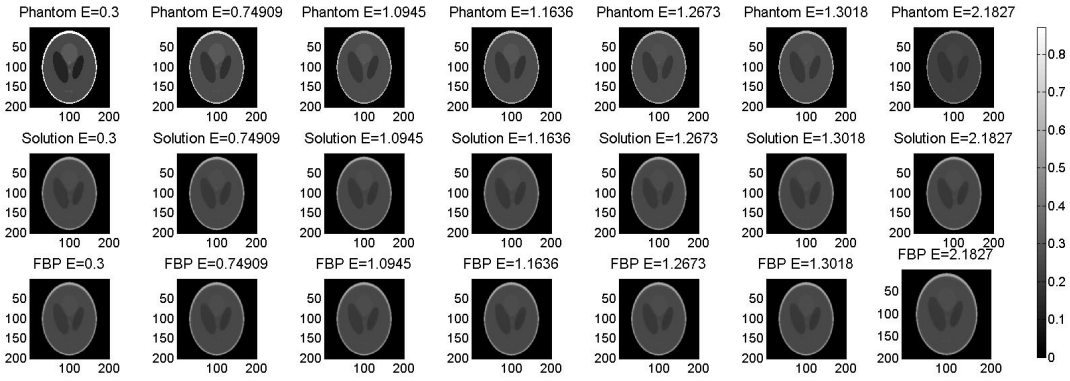


Figure 5.8: In this figure, we display the results for Example 2 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying no regularization. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new matrix-free polyenergetic nonlinear algorithm. The third row shows the starting guess (FBP solution) images which are the same for each energy level. It is clear that the results for our new algorithm are significantly better than the results for FBP.

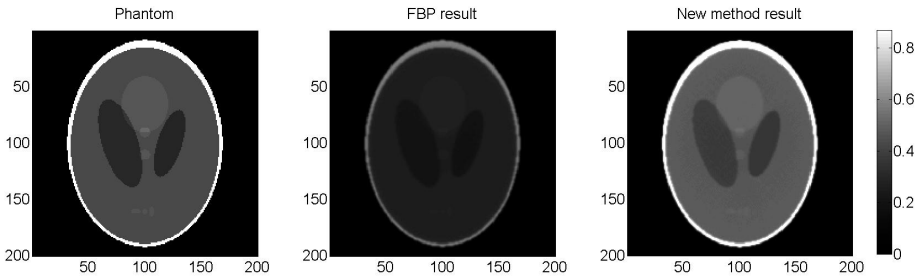
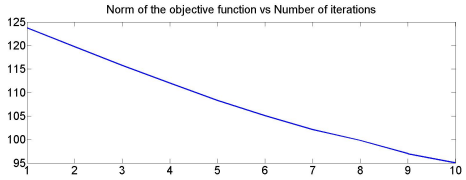
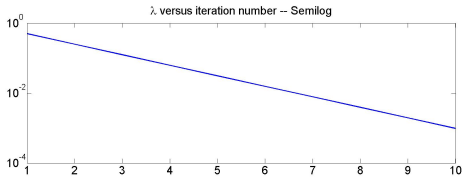


Figure 5.9: In this figure, we display the results for Example 2 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying no regularization. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8). It is clear that the results for our new algorithm are significantly better than the results for FBP.



(a) $\widehat{G}(\boldsymbol{\mu}^{(k)})$ versus the number of iterations for Example 2. We see that, after 10 iterations of our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm, $\widehat{G}(\boldsymbol{\mu}^{(k)})$ is smaller than for the FBP solution that we use as the starting guess for our new algorithm (compare the first and last points of the curve).



(b) $\lambda^{(k)}$ versus the number of iterations (see Equation (4.31))in Example 2.

Figure 5.10: In this figure, we display the results for Example 2 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying no regularization.

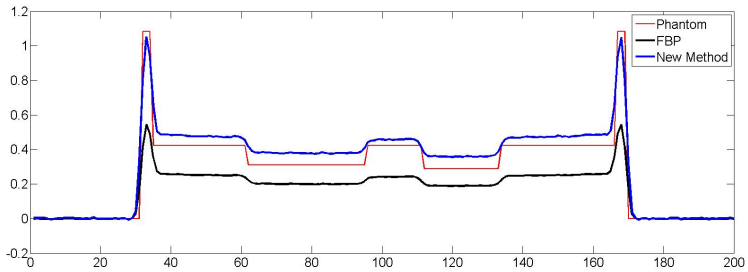


Figure 5.11: In this figure, we display the profile of the phantom, the FBP result and the solution of our new method for Example 2 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and no regularization. It is clear that the profile for our new method is much closer to the profile for the phantom than is the profile for FBP.

5.6.3 Example 3

In Example 3, we use the same amount of noise as in Example 2 ($C_{\text{noise}} = 0.5$), but, unlike Example 2, which uses no regularization, we apply Tikhonov regularization with the regularization parameter $\gamma = 10^{-3}$ in Example 3. The numerical results for Example 3 are presented in Figures 5.12, 5.13, 5.14, and 5.15.

In this experiment, the execution time for our new method is approximately 2561.03 seconds. We use the tolerance 10^{-3} and the maximum number of iteration of 10 for the stopping criteria for the LMF method. The tolerance used for CG at each LMF iteration is 10^{-1}. In this example we use 10 LMF iterations and the total number of CG iterations is 19.

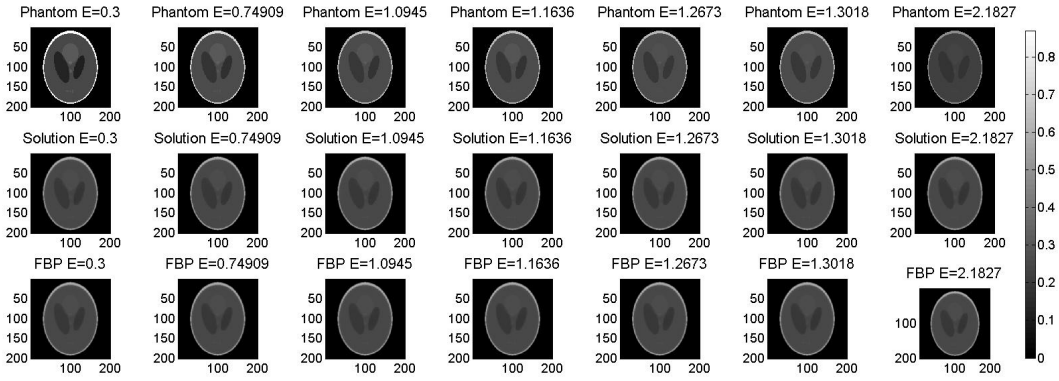


Figure 5.12: In this figure, we display the results for Example 3 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying Tikhonov regularization with $\gamma = 10^{-3}$. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new matrix-free polyenergetic nonlinear algorithm. The third row shows the starting guess (FBP solution) images which are the same for each energy level. It is clear that the results for our new algorithm are significantly better than the results for FBP.

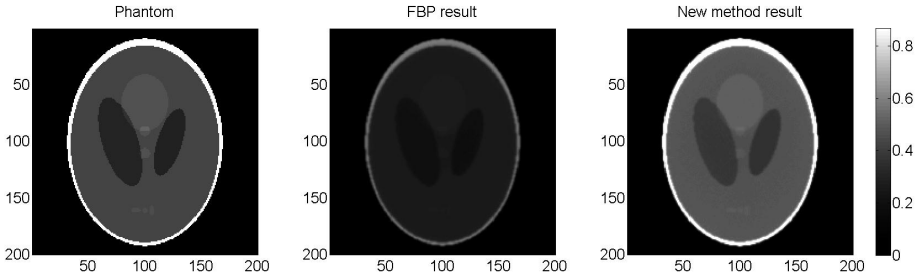
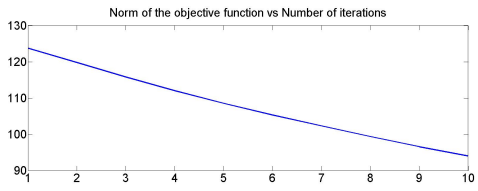
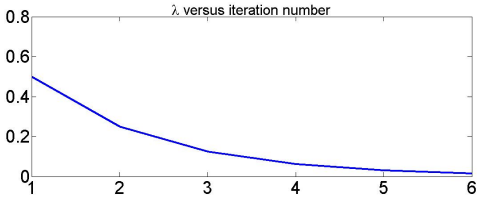


Figure 5.13: In this figure, we display the results for Example 3 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying Tikhonov regularization with $\gamma = 10^{-3}$. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8). It is clear that the results for our new algorithm are significantly better than the results for FBP.



(a) $\widehat{G}(\boldsymbol{\mu}^{(k)})$ versus the number of iterations for Example 3. We see that, after 10 iterations of our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm, $\widehat{G}(\boldsymbol{\mu}^{(k)})$ is much smaller than for the FBP solution that we use as the starting guess for our new algorithm (compare the first and last points of the curve).



(b) $\lambda^{(k)}$ versus the number of iterations (see Equation (4.31)).

Figure 5.14: In this figure, we display the results for Example 3 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and applying Tikhonov regularization with $\gamma = 10^{-3}$.

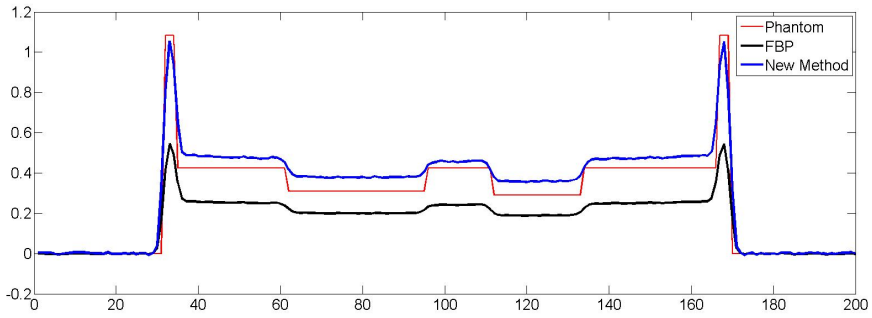


Figure 5.15: In this figure, we display the profile of the phantom, the FBP result and the solution of our new method for Example 3 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and Tikhonov regularization with $\gamma = 10^{-3}$. It is clear that the profile for our new method is much closer to the profile for the phantom than is the profile for FBP.

5.6.4 Example 4

Example 4 is similar to Example 3, except that, in Example 4, we regularize the objective function in the 1-norm with the regularization parameter $\gamma = 10^{-3}$, whereas, in Example 3, we apply Tikhonov regularization with the regularization parameter $\gamma = 10^{-3}$. Examples 3 and 4 use the same amount of noise ($C_{\text{noise}} = 0.5$). In Figures 5.16, 5.17, 5.18 and 5.19, we show the numerical results for Example 4.

In this experiment, the execution time for our new method is approximately 2585.16 seconds. We use the tolerance 10^{-3} and the maximum number of iteration of 10 for the stopping criteria for the LMF method. The tolerance used for CG at each LMF iteration is 10^{-1}. In this example, we use 10 LMF iterations and the total number of CG iterations is 20.

Note that the numerical results for Examples 3 and 4 are similar; both methods are equally effective for this reconstruction problem.

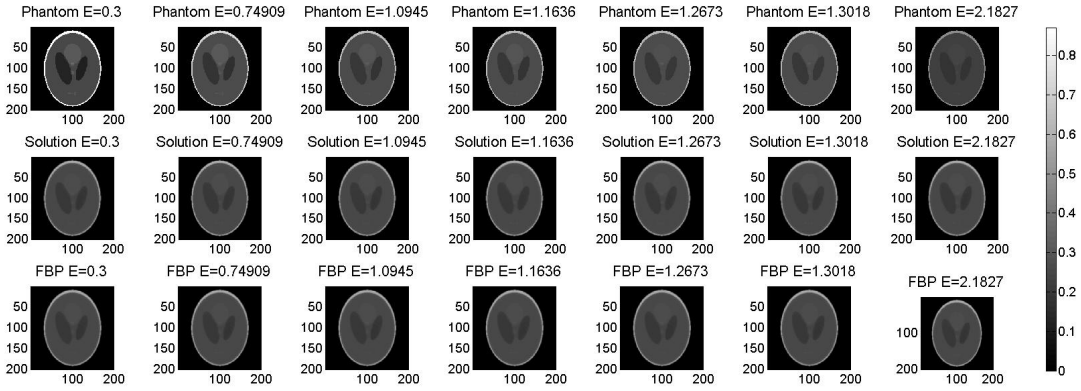


Figure 5.16: In this figure, we display the results for Example 4 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and regularizing in the 1-norm with $\gamma = 10^{-3}$. The first row contains different slices of the energy-dependent Shepp-Logan phantom, produced by the MATLAB function, `ephantom.m`, at different energy levels. The energy value at each level is reported on top of each image. The second row shows different slices of the reconstructed image computed by our new matrix-free polyenergetic nonlinear algorithm. The third row shows the starting guess (FBP solution) images which are the same for each energy level. It is clear that the results for our new algorithm are significantly better than the results for FBP.

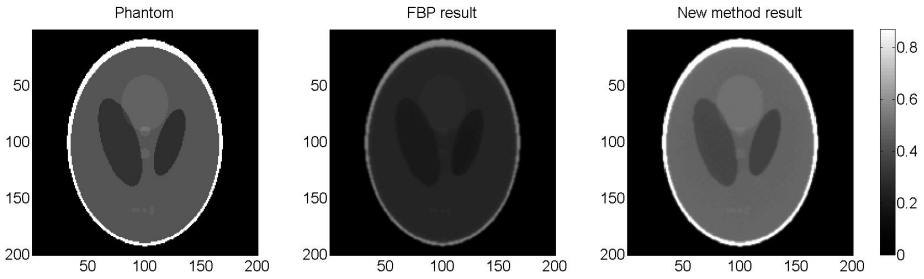
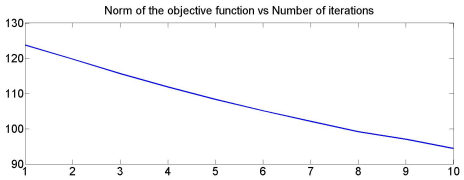
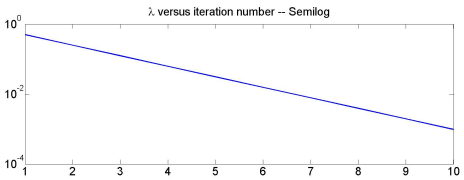


Figure 5.17: In this figure, we display the results for Example 4 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and regularizing in the 1-norm with $\gamma = 10^{-3}$. On the left, we show the original phantom data; in the middle, the FBP solution (we use the Hamming filter and linear interpolation in the `iradon` function in MATLAB); and on the right, we merge the results for our new method for different energy levels by using the discretized model (4.8). It is clear that the results for our new algorithm are significantly better than the results for FBP.



(a) $\widehat{G}(\boldsymbol{\mu}^{(k)})$ versus the number of iterations for Example 4. We see that, after 10 iterations of our new matrix-free LMF-based iterative polyenergetic reconstruction algorithm, $\widehat{G}(\boldsymbol{\mu}^{(k)})$ is much smaller than for the FBP solution that we use as the starting guess for our new algorithm (compare the first and last points of the curve).



(b) $\lambda^{(k)}$ versus the number of iterations (see Equation (4.31)).

Figure 5.18: In this figure, we display the results for Example 4 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and regularizing in the 1-norm with $\gamma = 10^{-3}$.

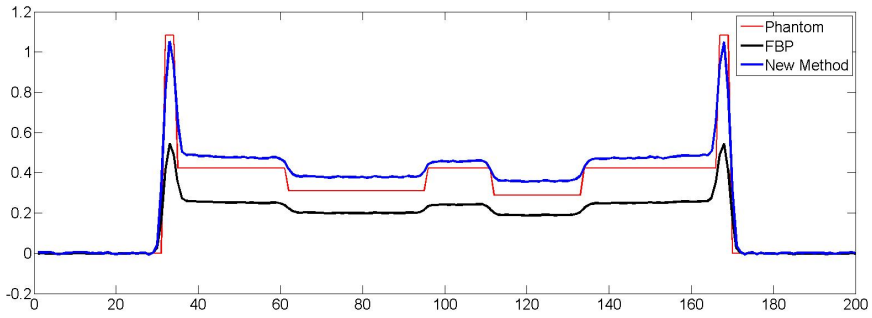


Figure 5.19: In this figure, we display the profile of the phantom, the FBP result and the solution of our new method for Example 4 with $N_{\text{grid}} = 200 \times 200$, $N_\varepsilon = 7$, $N_\theta = 180$, $N_t = 287$, $C_{\text{noise}} = 0.5$ and regularizing in the 1-norm with $\gamma = 10^{-3}$. It is clear that the profile for our new method is much closer to the profile for the phantom than is the profile for FBP.

Chapter 6

Conclusions and future work

6.1 Summary and conclusions

In this thesis, we first review the mathematical background needed for medical imaging in Chapter 2, including the Fourier transform and its inverse, the Radon transform, the Central slice theorem and the filtered back-projection (FBP) method, which can be used to compute the inverse of the Radon transform. We show how we can use this theory to build practical algorithms for different geometries, such as parallel-beam and cone-beam. In Chapter 2, we also introduce the matrix representation of the well-known FBP method. This representation helps establish a relation between the iterative methods, in particular ART and FBP. It also helps us develop the matrix-free polyenergetic iterative reconstruction algorithm in Chapter 5.

In Chapter 3, we review different families of iterative methods, such as the algebraic reconstruction techniques (ART) and the statistical image reconstruction techniques (SIRT). We explain that iterative methods can have several advantages over direct methods. First, they can incorporate some prior knowledge, including system geometry, detector response, object constraints, and they also permit modeling data noise. Second, an unrealistic assumption underlying FBP is that x-ray sources are monoenergetic. How-

ever, in practice, there is a nonuniform distribution of photons of different wavelengths that leads to a phenomenon physicists call beam hardening. Some iterative methods, such as SIRT, can model polyenergetic x-ray sources and thus account for beam hardening in the reconstruction. SIRT typically use a statistical model, such as a likelihood-based approach, to estimate the attenuation coefficient. In such techniques, to simplify the problem, some prior assumptions about the physical properties of the scanned object are often made. However, such prior assumptions can limit the scope and applications of the reconstruction method.

Based on the same continuous polyenergetic model used for SIRT, we develop a non-statistical iterative polyenergetic reconstruction method, in Chapter 4. Then we explain how we can use a quadrature rule (such as a Gaussian quadrature rule or the composite Trapezoidal rule) to discretize and approximate the outer integral in (4.1) with respect to different energy levels. We also show how we can take advantage of our knowledge about the Radon transform and the FBP method to discretize the energy-dependent continuous model with respect to different projections. We emphasize that in this model we account for the nonlinearity in the measurements. We study different solvers and nonlinear unconstrained optimization methods, such as a Newton-like method and an extension of the Levenberg-Marquardt-Fletcher (LMF) algorithm. We find that the extended LMF method produces better-quality images than the Newton-like method. Therefore, we apply the extended LMF method to solve our reconstruction problem. In this new reconstruction method, we discretize the polyenergetic physical model directly, with no prior assumption about the physical properties of the object of interest.

We show in our numerical experiments (using our energy-dependent mathematical Shepp-Logan phantom) that the LMF method is the most effective among the methods we have explored. However, if we store all the matrices in (4.31), the memory requirements are large and the computation time is long. In fact, in some test problems, we run out of memory. However, on the positive side, we report results with better image quality,

compared to FBP solutions, on small data sets.

In Chapter 5, to improve the efficiency of our approach, we use the known structure of the Radon matrix to better understand the structure of the Jacobian matrix of our objective function (see Figure 5.1). We also exploit some fast implementations of the Radon transform and the back-projection method to make our iterative polyenergetic method matrix-free and fast. The matrix representation of FBP introduced in Section 2.5 helps to understand this process better. Using our new reconstruction method, we get results with better image quality than FBP. On the other hand, FBP is much faster than our iterative reconstruction method. Nevertheless, our matrix-free approach is promising.

To improve the effectiveness of our new method, we regularize our optimization problem by the Tikhonov method and regularization in the 1-norm, using a difference kernel. As reported in Section 5.6, the results of both regularization methods are equivalently effective; the user may employ either technique, depending on the reconstruction problem. However, we emphasize that it is important to regularize the problem, since the problem is usually underdetermined. Therefore, without regularization, we may get results with poor image quality.

Finally, in Appendix A, we discuss our software package called `OSCaR` (Open Source Cone-beam Reconstructor). `OSCaR` uses the well-known and practical Feldkamp-Davis-Kress (FDK) algorithm, a filtered back-projection algorithm for cone-beam geometry. `OSCaR` is written in MATLAB and generates 3-D reconstructions from x-ray data acquired from cone-beam scanning geometries. The lack of practical, flexible, free FDK software implementations can hamper medical physics researchers and inhibit multi-institutional research collaboration. Recognizing the need for a common, referenceable imaging research software, we developed `OSCaR` as a research tool.

6.2 Future work

6.2.1 Using real data

To date, to test our new polyenergetic reconstruction algorithm, we have used mathematical phantoms with simulated beam hardening effects, and not "real" data from physical phantoms or human/animal subjects. To make our experimental results more convincing, we hope to apply our matrix-free polyenergetic nonlinear method to "real" sinogram data. Our long-term hope is that our algorithm, or an extension of it, will be used clinically.

6.2.2 Cone-beam geometry

The present algorithm assumes a two dimensional acquisition geometry, e.g., parallel-beam (see Section 2.4.3). However, we believe we can extend this algorithm to three dimensional geometries, such as cone-beam, which is described in Section 2.3.3.

6.2.3 Comparison with other polyenergetic techniques

In this thesis, we compare our numerical results for our new ployenergetic reconstruction algorithm to those of FBP. However, as discussed in Section 3.3, there are other iterative reconstruction techniques that are based on the same polyenergetic physical model (3.13) that we use. We hope to compare our polyenergetic reconstruction method to other available polyenergetic methods in the literature.

6.2.4 Reducing the running-time of our matrix-free polyenergetic reconstruction method

Long computation time is one of the main drawbacks of some reconstruction algorithms, iterative techniques in particular. This is acutely true of our new matrix-free polyener-

getic reconstruction method. As reported in Chapter 5, our new method is much slower than FBP.

However, in doing the final revision of this thesis, after the PhD oral, we noticed that we can improve the efficiency of our new method significantly. For example, our description of the computation of (5.12) in Section 5.4.1 uses $2N_\varepsilon$ matrix-vector multiplies with $\mathbf{R}$ and $2N_\varepsilon$ matrix-vector multiplies with $\mathbf{R}^T$. In our MATLAB program that we wrote to implement our new method, we use $2N_\varepsilon$ calls to the MATLAB function `radon` and $2N_\varepsilon$ calls to the MATLAB function `iradon` to implement the matrix-vector multiply (5.12) at each CG iteration. However, in doing the final revision of this thesis, we noticed that the matrix-vector multiply (5.12) can be implemented with N_ε calls to `radon` and N_ε calls to `iradon`. We also found similar savings in other parts of our code that should speed up our MATLAB implementation of our new matrix-free polyenergetic reconstruction method significantly. We plan to investigate this further and report the results in a forthcoming paper.

Moreover, if we can make the computations parallel, then they may be much faster. We plan to investigate the parallelization of our new matrix-free polyenergetic reconstruction method, and possibly run them on parallel graphics processing units (GPUs). There are several GPU libraries, such as OPENGL, CG and CUDA, available at the moment. Moreover, software libraries for GPUs are growing very fast. Therefore, we believe that, in the near future, it should be possible to build an effective, parallel, GPU-based implementation of our algorithm. For more information about GPU programming, refer to [1, 45].

Appendix A

OSCaR

`OSCaR` (Open Source Cone-beam Reconstructor) [49, 50] is a software package written in MATLAB developed for generating 3-D reconstructions from x-ray data acquired from cone-beam scanning geometries. It is based on the Feldkamp-Davis-Kress (FDK) algorithm [21], for 3-D cone-beam CT (CBCT) reconstruction.

In this appendix, we describe version 3 of `OSCaR`, the current version of `OSCaR` since 2010. The latest version of `OSCaR`, as well as the associated documentation and examples, can be obtained from the website `http://www.cs.toronto.edu/~nrezvani/OSCaR.html`.

The lack of practical, flexible, free FDK software implementations can hamper medical physics researchers and inhibit multi-institutional research collaboration. Recognizing the need for common, referenceable imaging research software, `OSCaR` was developed as a research tool for free distribution by the American Association for Physicists in Medicine (AAPM). The goal is to offer a straightforward, open source code and GUI (Graphical User Interface) implementation that emphasizes flexibility, generality, and ease of use. The implementation has a transparent interface for 3-D image reconstruction with the intention of providing a useful base platform for developing advanced techniques (e.g., artifact correction) or for conducting image quality studies (e.g., selection of optimal

reconstruction filters). The code is intended for research use, not clinical or commercial implementation. Although compiled software is certainly faster than interpreted Matlab code, a Matlab implementation circumvents the necessity of custom compiled libraries. The intention is not to produce the fastest implementation; rather, it is to provide code from which researchers can start developing their own algorithms.

Broadly speaking, `OSCaR` has three main components: pre-processing, reconstruction, and export. In the pre-processing stage, CBCT data is parsed from a broad, general base of standard data-file formats, accessible to Matlab (e.g., DICOM[1], binary, JPEG, TIFF, PNG, etc.). The source-detector pair is assumed to move in a circular path, but `OSCaR` can allow for variation in the position of the piercing point as a function of the projection angle. Parameters associated with geometric corrections, sampling, air normalization, and other device-dependent properties that affect acquisition of the sinogram data can be specified using the pre-processing GUI prior to reconstruction.

In the reconstruction stage, `OSCaR` permits the specification of the Field-Of-View (FOV), voxel size and reconstruction filters. The actual voxel-driven reconstruction uses the well-known FDK filtered back-projection algorithm. Then, in the export stage, the reconstructed data can be saved as a three dimensional matrix, which can be exported as a standard data-file, such as DICOM, binary, etc, and visualized.

We hope that `OSCaR` will soon be made freely available on the AAPM web-site to members of the AAPM for research in algorithm development, CBCT image quality studies, and multi-institutional collaboration.

[1]DICOM (Digital Imaging and Communications in Medicine) is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a network communications protocol.

A.1 A reconstruction algorithm for cone-beam machines

As discussed in Section 2.4.3, a view for a parallel-beam scanner produces measurements of $\mathcal{R}\mu$ from a family of parallel lines. Therefore, the Central slice theorem (Theorem 2.2.1) applies to give an approximation to the Fourier transform of the attenuation coefficient along lines passing through the origin. However, cone-beam geometry is more complicated, since the samples of $\mathcal{R}\mu$ are associated with a set of non-parallel lines that form a cone shape and all the lines pass through a common point. Hence, the Central slice theorem is not directly applicable. There are two general approaches to reconstructing images from the data collected by a cone-beam machine (this is also true for fan-beam machines). We could sort and interpolate the cone-beam data to obtain data appropriate for a fan-beam (or in some cases parallel-beam) algorithm. Alternatively, we could extend a parallel-beam or fan-beam algorithm to cone-beam geometry. The Feldkamp-Davis-Kress (FDK) algorithm [21] reviewed below is such an algorithm.

The FDK algorithm was introduced in 1984 and since then it has been the most commonly used algorithm for 3-D reconstruction (there are many commercial scanners that use variants of this algorithm). In the same year, a similar algorithm for divergent beams was introduced by Webb [59].

We assume a planar detector, the same assumption as the original form of FDK. The reconstruction is based on filtered back-projection similar to the two-dimensional algorithm discussed previously. Let (u, v) be the co-ordinates of the detector, (x, y, z) the co-ordinates of the real world, D the source-axis distance, g the filter and β the projection angle (see Figure A.1). Using the FDK formulation, we get

$$\mu(x, y, z) \approx \int_0^{2\pi} \frac{D^2}{U^2} [\mathfrak{t}(\mathcal{R}\mu) * g](\beta, u', v')\, d\beta, \tag{A.1}$$

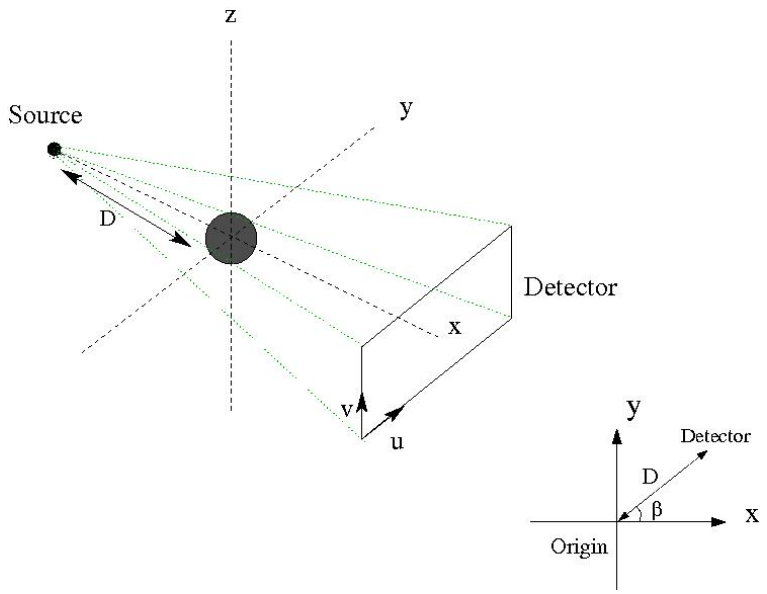


Figure A.1: Cone-beam projection.

where

$$\mathfrak{t}(\mathcal{R}\mu)(\beta,u,v)=\frac{D}{\sqrt{D^2+u^2+v^2}}[\mathcal{R}\mu](\beta,u,v), \tag{A.2}$$

$$U=D+x\cos\beta+y\sin\beta, \tag{A.3}$$

$$u'=\frac{D}{U}(x\sin\beta+y\sin\beta), \tag{A.4}$$

$$v'=\frac{D}{U}z. \tag{A.5}$$

In the discrete case, the integral is replaced by a sum over the projection angles. Then we need a two-dimensional interpolation of the filtered projection data for each term of the back-projection sum. For more details about FDK reconstruction, refer to [21, 51, 55].

A.2 Assumptions and limitations

The following assumptions and general recommendations are pertinent to the use and further development of `OSCaR`. The geometry for acquisition of the 3-D sinogram is circular (i.e., the source-axis-distance and the source-detector-distance are constant). Also,

no detector tilt is allowed. However, the code does allow variations in the location of piercing points (u_0, v_0) from view to view.

In CBCT, the projection geometry involves a point x-ray source with a divergent x-ray beam. We assume that the detector is flat and has rectangular detector elements of uniform dimensions. The projections may be collected at nonuniform projection angles. The measurement distances are in centimeters unless otherwise stated. The origin of a pixel is its centre.

`OSCaR` handles projection sets covering exactly 180 + fan or 360 degrees. Anything in between is improperly weighted. The code has been tested on Matlab 2009a with the Image Processing Toolbox 6.2. Potential file formats for input projection data can be any image file format accessible to Matlab (e.g., DICOM, jpg, tiff, etc). Alternatively, all projections can be input in a single binary data file. Sinogram data stored in binary files are ordered in column major order (consistent with the Matlab convention). Loading/saving data from three-dimensional arrays can take some time depending on the memory available on the computer and disk-access time.

The largest 3-D sinogram that can be loaded by `OSCaR` depends on the amount of memory available on the computer. The entire 3-D sinogram must fit into memory with pixel values stored as double precision floating point numbers. Given that a single projection of 192 by 256 pixels needs about 400 KB of memory, 300 projections require about 113 MB. Conversely, 1 GB of free accessible memory corresponds to 320 (192×256 pixel) projections. Similar restrictions apply to the memory required for reconstructions; the user must be able to store at least double the memory required for the user-specified reconstruction array (for work arrays and the like).

In the next sections, we describe how to run `OSCaR`.

A.3 Running OSCaR

Prior to launching OSCaR, the directory containing the source code for OSCaR must be in your Matlab path. Also, a .csv (*comma-separated value*, see Section A.3.2) file needs to be prepared, and stored in the same directory that contains the 3-D sinogram projections. The requirements are outlined in Section A.3.2 below.

The user has two options for using OSCaR. The first option is invoked by calling OSCaRMain which launches a GUI. The GUI OSCaRMain provides access to two other GUIs, OSCaRPreprocess and OSCaRReconstruct. The second option for using OSCaR is entirely command-line based and is invoked using OSCaR with appropriate inputs.

A.3.1 The GUI OSCaRMain

OSCaRMain (shown in Figure A.2) is the primary GUI. The buttons can be used to launch OSCaRPreprocess and OSCaRReconstruct or to cancel the computation.

A.3.2 The GUI OSCaRPreprocess

The GUI OSCaRPreprocess (shown in Figure A.3) is used to input the 3-D sinogram data for processing prior to reconstruction. The user can initiate the GUI OSCaRPreprocess either by clicking the button *Load Projections* from the GUI OSCaRMain, or by entering OSCaRPreprocess at the Matlab command prompt. The individual projections are assumed to have been captured at N_{proj} different gantry angles. Each projection consists of an image of resolution $N_u \times N_v$ pixels. The v-direction on the detector is assumed to be parallel to the axis of revolution of the source-detector pair; the u-direction on the detector is assumed to be perpendicular to the v-direction.

Prior to loading the $N_u \times N_v \times N_{\text{proj}}$ values comprising the 3-D sinogram from disk, the user must provide some information about how the data was collected and how the data is stored. Although some of it can be entered into the GUI OSCaRPreprocess directly, some

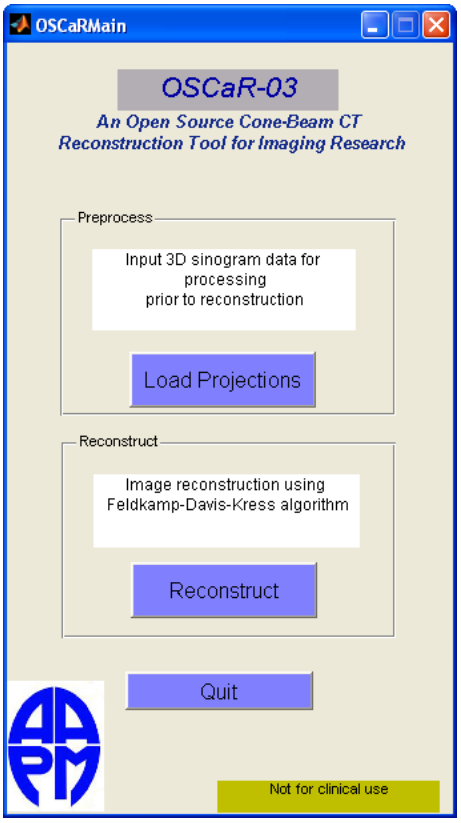


Figure A.2: `OSCaRMain`

of this information can vary for each of the N_{proj} projections. As such, before loading the 3-D sinogram into memory, `OSCaRPreprocess` requires the following information from the user: the geometry/resolution parameters (pertaining to all the projections); storage parameters (pertaining to all the projections); and the projection-dependent parameters. The details of how the user can provide this information follows.

Geometry/resolution parameters

These are the parameters that describe the geometry and resolution of the cone-beam apparatus used to acquire the projection data. These parameters are summarized as follows:

du	Pixel length (cm) in u direction (perpendicular to axis)
dv	Pixel length (cm) in v direction (parallel to axis)
SAD	Source-Axis-Distance (cm)
SDD	Source-Detector-Distance (cm)

.

The GUI `OSCaRPreprocess` has textboxes on the right-hand side that allow users to specify each of du, dv, SAD, and SDD appropriately.

Storage parameters

The projection data are assumed to be stored in one of two generic ways: either the N_{proj} projections are all stored in a single binary data file in column major order or the projections are stored in N_{proj} individual image files, each of which is in some format that Matlab can import (e.g., JPEG, TIFF, DICOM, etc.).

The GUI `OSCaRPreprocess` has a pair of radio-buttons at the top left corner (*Binary File* versus *Other Formats*, see Figure A.3) to specify the choice of data format.

If the user selects *Binary File*, *all* the N_{proj} projections are assumed to be stored in a single file in column-major order (the filename should be provided in the `.csv` file, see Section A.3.2). As such, the user must edit the textboxes to provide the *Row No.* (i.e., N_u), the *Column No.* (i.e., N_v), and the *Projection No.* (i.e., N_{proj}). The user also needs to specify, using pull-down menus, the *Pixel Type* (`int32`, `float32`, etc.) and the *Machine Format* required for reading the binary data. Strings describing the machine formats are summarized below.

`n`	Numeric format of machine on which MATLAB is running (default)
`b`	32 bit IEEE floats with big-endian byte ordering
`l`	32 bit IEEE floats with little-endian byte ordering
`s`	64 bit IEEE floats with big-endian byte ordering
`a`	64 bit IEEE floats with little-endian byte ordering
`d`	VAX D floats and VAX ordering
`g`	VAX G floats and VAX ordering

If the user selects *Other Formats*, the N_{proj} projections are assumed to be stored in N_{proj} *individual image files* (whose names are provided in the `.csv` file, see Section A.3.2). In this case, the values of N_u and N_v can be inferred directly from the image files as they are loaded (recall that the values of N_u and N_v must be the same for every projection).

After input parameters N_u, N_v, N_{proj} and the input file format are specified, an estimate of the amount of memory required is displayed in the box "Approximate Required Memory (RAM)", on the right side of the GUI. The user can check whether the machine has enough memory to run `OSCaRPreprocess`. For example, to process `SampleData.csv` and `20070611-1-roc-750434170.img`, use the following values:

N_{proj}	320
N_{row}	192
N_{col}	256
`du`	0.1552
`dv`	0.1552
`SAD`	100
`SDD`	155

Projection-dependent parameters

Certain parameters can vary from projection to projection. These should be specified by the user in a `.csv` (comma separated values) file to avoid having to enter each parameter

into a textbox in the GUI `OSCaRPreprocess`. A `.csv` file can be prepared and exported using most commonly-used spreadsheet programs. The user is prompted for the name of the `.csv` file before the 3-D sinogram can be loaded into memory.

The `.csv` file should consist of N_{proj} rows, each with 6 columns. The 6 columns of the kth row of the `.csv` file consist of information pertinent to the kth projection ($k = 1 \colon N_{\text{proj}}$). These columns are as follows:

$$\text{filename} \;\Big|\; \theta_G \;\Big|\; u_{\text{off}} \;\Big|\; v_{\text{off}} \;\Big|\; I_0 \;\Big|\; w \;\; .$$

filename	string naming file in which kth projection is stored
θ_G	gantry angle of kth projection (degrees)
u_{off}	offset of centre of detector perpendicular to axis (cm)
v_{off}	offset of centre of detector parallel to axis (cm)
I_0	air normalization (same units as values in projections)
w	weight (dimensionless)

.

An example of such a `.csv` file is included in this package.

Orientation buttons

In the *orientation* box on the right side of the GUI, three buttons, *Flip Horizontally*, *Flip Vertically*, and *Transpose*, can be found. Once a suitable orientation is found, the modified data is displayed in the GUI, and the user can save the projection data and export it to the reconstruction component, `OSCaRReconstruct`.

Export to `OSCaRReconstruct`

When the user has completed the preprocessing stage described above, the data can be exported to a `.mat` file[2] that can be used in the reconstruction stage (`OSCaRReconstruct`). The `.mat` file consists of a structure, `experiment`, with two fields: `param` and

[2] A `.mat` file is a file which contains saved variables written in a MATLAB specific format.

`data`. The first field, in turn, consists of fourteen subfields that are filled by the values displayed in the GUI `OSCaRPreprocess`; the second one, as its name suggests, contains the data from the scanner.

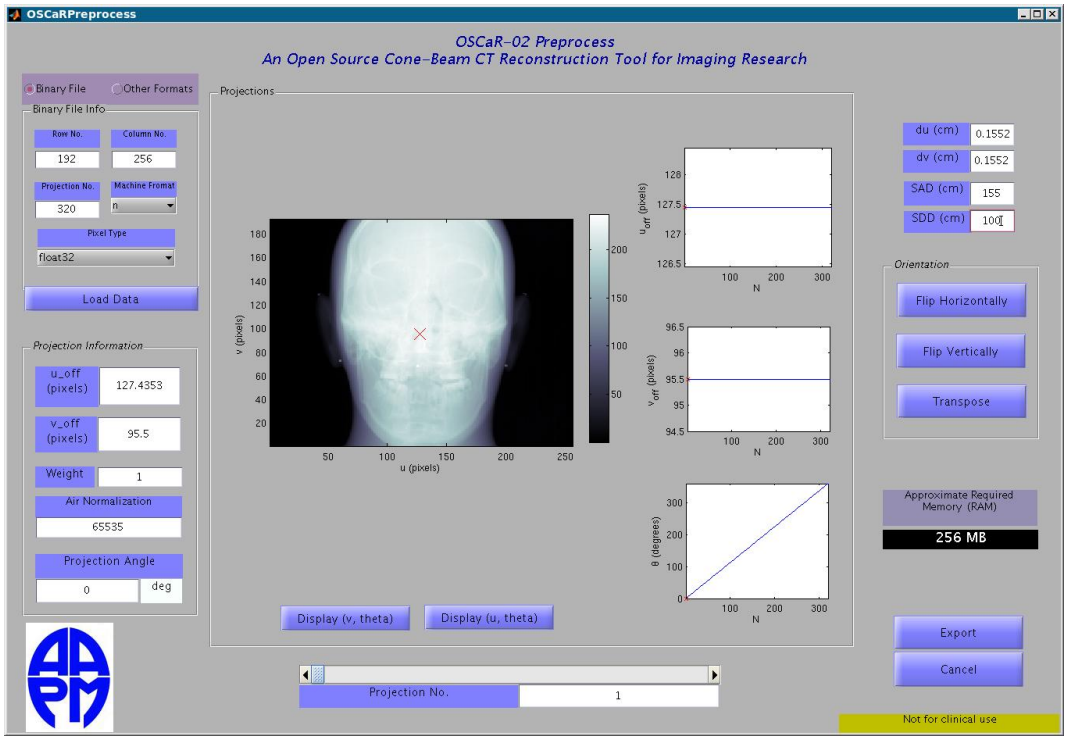


Figure A.3: `OSCaRPreprocess`

A.3.3 The GUI `OSCaRReconstruct`

The GUI `OSCaRReconstruct` (shown in Figure A.4) is used for image reconstruction. The user can initiate the GUI `OSCaRReconstruct`

- by clicking the button *Reconstruct* from the GUI `OSCaRMain`, or
- by entering `OSCaRReconstruct` at the MATLAB command prompt.

Once `OSCaRReconstruct` is launched, the user must load the `.mat` file produced in the previous stage by `OSCaRPreprocess`.

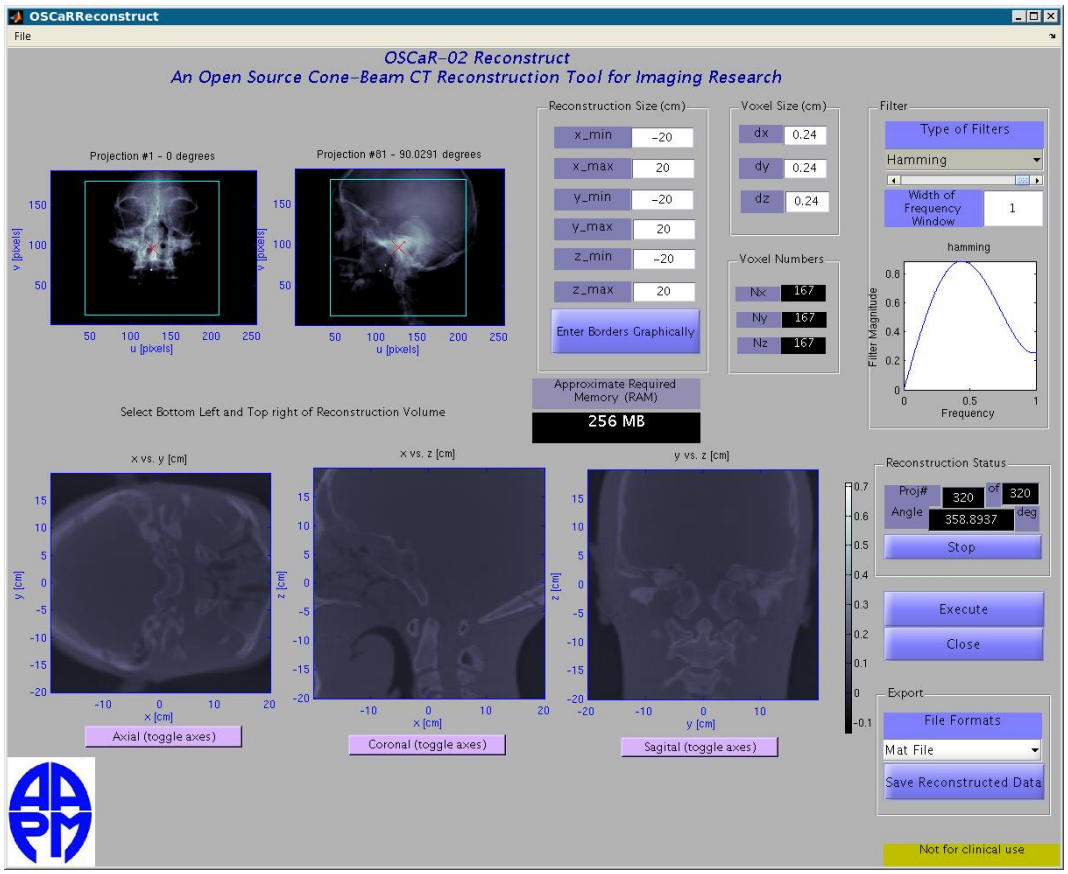


Figure A.4: `OSCaRReconstruct`

Reconstruction size

The user is required to specify the box $[x_{min}, x_{max}], [y_{min}, y_{max}]$ and $[z_{min}, z_{max}]$ that bounds the voxelized reconstruction either graphically or numerically.

1. Graphically: After pushing the button *Enter the Borders Graphically*, the user must click on the images to select the bottom left and top right of the reconstruction volume. (In total, two points on each image are required.)

2. Numerically: The user must enter the borders numerically.

These values provide the dimensions of the reconstruction domain in the real world. If the user enters the reconstruction sizes incorrectly (e.g., out of range or the minimum value is greater than the maximum value), an error message appears. The default values

yield a two dimensional axial reconstructed slice. An estimate of the amount of memory needed for reconstruction is displayed in a box near the centre of the GUI with the title "Approximate Required Memory (RAM)". The user should verify that the machine has enough memory to run `OSCaRReconstruct`.

Filter

The next step is selecting the appropriate filter with a desired width of frequency window (between 0 and 1). On the right side of the GUI `OSCaRReconstruct`, a drop down menu offers a selection of different filters, such as Ram-Lak, Shepp-Logan, Cosine, Hamming, and Hann. These filters are defined in the function `OSCaRFilter.m`. This function returns a vector of coefficients that is multiplied element-wise by the rows of the Fourier transform of the sinogram data. This vector of coefficients is the filter. The input arguments of `OSCaRFilter` are the length and the spatial resolution of the projection data, and the fraction of frequencies below the Nyquist frequency which we want to pass. The latter is a value between zero and one. The user must enter the width of the frequency either in the edit box or by using the slider on the right side of the `OSCaRReconstruct` GUI.

In `OSCaRFilter` the filters are defined as a multiplication of the ramp filter by the following apodizing windows:

- Shepp-Logan: $\mathrm{sinc}(x)$,
- Cosine: $\cos(x)$,
- Hamming: $0.54 + 0.46\cos(x)$,
- Hann: $0.5 + 0.5\cos(x)$.

Alternatively, the user may choose to enter their own apodizing window, by choosing "new filter" in the drop down menu on the right. After selecting this option, the user is asked to enter the apodizing window as a function of x. Note that here x is a

vector. As an example, the user can enter $\cos(x)$ or $\cos(x) .* \sin(x)$. (Note the array element-wise multiplication operator ".*", since x is a vector.) If the function is not recognized by Matlab, an error message alerts the user; otherwise, the corresponding filter (the apodizing window multiplied by the ramp filter) is displayed and is used in the reconstruction. If the user chooses "No Filter", then the function returns an empty vector.

In the filtering part of `OSCaR`, scaling of time and frequency domains are computed according to reciprocity relations as in [8].

Execute and export

Pressing the "Execute" button on the right side of the GUI starts the reconstruction process using the function `OSCaRFDK.m`. This function uses the Feldkamp-Davis-Kress (FDK) algorithm [21, 59].

After fixing the projection angle, `OSCaRFDK` returns the value calculated by the FDK algorithm for each voxel. Then the program loops over the projection angles, and the calculated values by `OSCaRFDK` are summed together. The `OSCaRFDK` routine uses the *nearest neighbour interpolation* to find the detector coordinates (u, v) that are closest to the projections of voxels (x, y, z). Note that, we have vectorized the FDK algorithm to increase the computational speed.

When the reconstruction is complete, the user can save the reconstructed data and the parameters defining the FOV (Field-of-View) of the reconstructed volume in three ways:

1. Save the reconstructed data in a `.mat` file. The user can specify a name for the `.mat` file, but the reconstructed data (P), and the parameters defining the FOV of the reconstructed volume, $(xmin, xmax, ymin, ymax, zmin, zmax)$, are saved in a structure called *Save_data*, with seven fields. To open the `.mat` file (reconstructed data), the user can use the following command in Matlab.

```
>> load('NameofMatFile.mat');
>> Save_data


Save_data =


       P: [167x167x167 double]
    xmax: 20
    xmin: -20
    ymax: 20
    ymin: -20
    zmax: 20
    zmin: -20
```

2. Save the reconstructed data as a binary file and the needed parameters, such as the reconstruction borders and voxel sizes, in a text file.

3. Save the reconstructed data as DICOM files.

A.3.4 The function `OSCaR`

This function opens a `.mat` file that stores preprocessed projection data as created by `OSCaRPreprocess`. The function `OSCaR` can then be used to perform the FDK reconstruction. By setting the value of `verbose` to 1 or 0, the user may choose to have the reconstructed volumes (axial, sagittal, and coronal) displayed, or not, respectively. The function `OSCaR` returns the reconstruction as its output.

Syntax

```
OSCaR(FileName,'PropertyName',PropertyValue)
```

where,

- **FileName**:: String.

  "FileName" is the name of the `.mat` file to be loaded as built by `OSCaRPreprocess`. This file contains a structure named *experiment*, with two fields, *param* and *data*. The first field has the subfields: $proj_{filenames}$, θ, u_{off}, v_{off}, I_0, weights, N_{proj}, N_{row}, N_{col}, du, dv, SAD, SDD, and IAD, and the second field has the projection data.

- **PropertyName** and **PropertyValue**:

PropertyName	PropertyValue	Default
FilterName	'No filter','hamming', 'hann', 'cosine', 'shepp-logan', 'ram-lak'	No filter
ReconstructionVolume	Vector of length 6 (xMin,xMax,yMin, ... , yMax,zMin,zMax)	Entered Graphically
VoxelSize	Vector of length 3 (dx,dy,dz)	Determined Internally by using a geometry-dependent formula
FilterWidth	Width of Frequency Window (Real value between 0 and 1)	1
Verbose	0 or 1	1

Note that to enter the borders graphically, the user must select bottom left and top right corners of the reconstruction volume.

A.4 One quick example

Assuming the Matlab path is configured appropriately and the data is in the working directory, the buttons in the GUI `OSCaRMain`, *Load Projections* and *Reconstruct*, can be

used to initiate `OSCaRPreprocess` and `OSCaRReconstruct`, respectively.

1. Enter `OSCaRMain` at the Matlab command prompt to launch the main GUI.

2. Launch `OSCaRPreprocess` by pressing the button *Load Projections* in the `OSCaRMain` GUI.

3. Select the radio-button *Binary File* (as opposed to *Other Formats*) from the top-right corner of the GUI `OSCaRPreprocess`.

4. When prompted, enter the following values: N_{proj}=320, N_{row}=192, N_{col}=256, `Machine Format`= n and `Pixel Type`= `float32`.

5. Press *Load Data* and open `SampleData.csv` and `20070611-1-proc-750434170.img` from the `examples` folder. The projection images are now displayed. The user can fix the orientation of the images by using the buttons on the left side, *Flip Horizontally*, *Flip Vertically*, and *Transpose*. In this case, use *Transpose* and *Flip Vertically*.

6. Enter the following data into the textboxes on the right side of the GUI: `du`=0.1552, `dv`=0.1552, `SAD`=100 and `SDD`=155

7. Press the *Export* button and name the exported file `SampleData.mat`. All the preprocessed projection data together with the acquisition parameters are now saved in the `.mat` file `SampleData.mat`.

8. Launch the GUI `OSCaRReconstruct` using the button in the GUI `OSCaRMain`.

9. When prompted, open the file `SampleData.mat` produced by `OSCaRPreprocess`.

10. Input the reconstruction volume graphically:

    - Click *Enter the Borders Graphically*.
    - Select the bottom left and top right of a rectangle in the left image.

- Select the bottom left and top right of a rectangle in the right image.

Values can be adjusted using the text-box *Reconstruction Size*.

11. Select a filter (e.g., `hamming`) from the pull-down menu on the right of the GUI `OSCaRReconstruct`.

12. Select the desired width of frequency window (between 0 and 1) using the slider in the upper right corner of the GUI `OSCaRReconstruct`.

13. Press the *Execute* button to start the reconstruction. *This may take some time to complete.*

14. When finished, you will be prompted to save the reconstruction (either as a `.mat` file, a binary file, or a DICOM file).

As an alternative to using `OSCaRReconstruct` in step (8) above, the function `OSCaR` can be used to carry out the reconstruction from pre-processed data as created by `OSCaR-Preprocess`. For example, consider entering the following at the MATLAB prompt:

```
M = OSCaR('examples/SampleData.mat', ...
          'FilterName', 'hamming', ...
          'ReconstructionVolume', ...
          [-15,15,-15,15,0,0], 'Verbose',0)
```

The output `M` is a MATLAB structure whose fields contain the data describing the reconstruction. More explicit details are available in the User Guide [49].

A.5 Standalone executable

In the `OSCaR` package, we have also included a Windows executable version of the GUIs called `OSCaREXE`. The standalone application `OSCaREXE` is generated by the MATLAB

compiler and can be found in the exe folder. The application `OSCaREXE` can be run on any Windows machine even if Matlab is not installed.

The user must ensure that the correct version of the Matlab Compiler Runtime (MCR) environment is installed on the target machine. We have used Matlab 2008b, which uses MCR version 7.9. The MCR installer is included in this package in the exe folder. The user may already have the MCR installed on their machine. To verify the version number of the installed MCR, type the following command at the Matlab command prompt:

```
[mcrmajor, mcrminor] = mcrversion
```

If the user does not have Matlab or the correct version of MCR installed on their computer, then they can run `MCRInstaller` in the exe folder.

After installing the MCR, to run the standalone executable, the user must launch `OSCaREXE` in the exe folder. This loads the GUI `OSCaRMain` and the user can proceed to follow the steps described in the previous sections.

Bibliography

[1] *GPGPU: General-purpose computation on Graphics Processing Units.* `http://gpgpu.org/`.

[2] A. H. Andersen and A. C. Kak. Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm. *Ultrasonic Imaging, Elsevier*, 6(1):81–94, 1984.

[3] C. H. Atkinson and J. Soria. Algebraic reconstruction techniques for tomographic particle image velocimetry. *16th Australasian Fluid Mechanics Conference*, pages 191–198, 2007.

[4] M. Balda. Levenberg-Marquardt-Fletcher algorithm for nonlinear least-squares problems. *MathWorks, File Exchange, ID 16063*, 2007. `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=16063`.

[5] Jacob Beutel, Harold Kundel, Richard L. Van Metter, and Milan Sonka. *Handbook of Medical Imaging: Medical Image Processing and Analysis*, volume 1. Physics and Psychophysics, SPIE Press, 2000.

[6] J. R. Bilbao-Castro, J. M. Carazo, J. J. Fernanndez, and I. Garcia. Parallel, distributed and network-based processing. *Euromicro Conference*, 2004.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

[8] W. L. Briggs and V. E. Henson. *The DFT: An Owners' Manual for the Discrete Fourier Transform*. SIAM, 1995.

[9] R. A. Brooks and G. Di Chiro. Beam hardening in x-ray reconstructive tomography. *Physics in Medicine and Biology*, 21(3):390–398, 1976.

[10] Charles Byrne. *Iterative Algorithms in Tomography*. Department of Mathematical Sciences, University of Massachusetts Lowell, 2005. `http://faculty.uml.edu/cbyrne/nlat.pdf`.

[11] J. F. Canny. Finding edges and lines in images. *Massachusetts Institute of Technology Artificial Intelligence Lab Technical Report*, 1(2), 1983.

[12] J. F. Canny. A computational approach to edge detection. *IEEE Transactions, Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[13] Ho-Shiang Chueh, Wen-Kai Tsai, Chih-Chieh Chang, Shu-Ming Chang, Kuan-Hao Su, and Jyh-Cheng Chen. Development of novel statistical reconstruction algorithms for poly-energetic x-ray computed tomography. *Computer Methods and Programs in Biomedicine*, 92(3):289–293, 2008.

[14] T. F. Coleman and Y. Li. On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds. *Mathematical Programming*, 67(2):189–224, 1994.

[15] T. F. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.

[16] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaa, and L. E. Meester. *A Modern Introduction to Probability and Statistics*. Springer, 2005.

[17] Idris A. Elbakri. *Statistical reconstruction algorithms for polyenergetic x-ray computed tomography*. PhD thesis, Department of Electrical Engineering: Systems, University of Michigan, 2003.

[18] Idris A. Elbakri and Jeffrey A. Fessler. Statistical image reconstruction for polyenergetic x-ray computed tomography. *IEEE Transactions on Medical Imaging*, 21(2):89–99, 2002.

[19] Charles L. Epstein. *Introduction to the Mathematics of Medical Imaging*. Prentice Hall, Upper Saddle River, N.J., 2003.

[20] Hakan Erdogan and Jeffrey A. Fessler. Monotonic algorithms for transmission tomography. *IEEE Transactions on Medical Imaging*, 18:801–814, 1999.

[21] L. A. Feldkamp, L. C. Davis, and J. W. Kress. Practical cone-beam algorithm. *Journal of the Optical Society of America A*, 1(6):612–619, 1984.

[22] R. Fletcher. A modified Marquardt subroutine for nonlinear least squares. *Harwell*, 1971.

[23] R. Fletcher. *Practical Methods of Optimization*. Wiley-Interscience New York, New York, USA, 1987.

[24] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal Numerical Analysis*, 17:238–246, 1980.

[25] R. Gordon, R. Bender, and G. T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography. *Journal of Theoretical Biology*, 29:471–482, 1970.

[26] Irina F. Gorodnitsky and Bhaskar D. Rao. A recursive weighted minimum norm algorithm: analysis and applications. *Proceedings of the 1993 IEEE international*

conference on acoustics, speech, and signal processing: digital speech processing - Volume III, pages 456–459, 1993.

[27] R. W. Hamming. *Digital filters*. Prentice Hall, Englewood Cliffs, N.J., 1989.

[28] Amy K. Hara, Robert G. Paden, Alvin C. Silva, Jennifer L. Kujak, Holly J. Lawder, and William Pavlicek. Iterative reconstruction technique for reducing body radiation dose at CT: feasibility study. *American Journal of Roentgenology*, 193(3):764–771, 2009.

[29] Gabor T. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Media, Springer, New York, 2009.

[30] Gabor T. Herman and Lorraine B. Meyer. Algebraic reconstruction techniques can be made computationally efficient. *IEEE Transactions on Medical Imaging*, 12(3):600–609, 1993.

[31] M. Jiang and G. Wang. Convergence of the simultaneous algebraic reconstruction technique (SART). *IEEE Transactions on Image Processing*, 12(8):957–61, 2003.

[32] Peter M. Joseph and Robin D. Spital. A method for correcting bone induced artifacts in computed tomography scanners. *Computer Assisted Tomography*, 2(1):100–108, 1978.

[33] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Prentice-Hall, Upper Saddle River, NJ, USA, 1998.

[34] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. SIAM, Philadelphia, PA, 2001.

[35] K. Levenberg. A method for the solution of certain problems in least-squares. *Quarterly Applied Mathematics*, 2:164–168, 1944.

[36] Jae S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice-Hall, Upper Saddle River, NJ, USA, 1990.

[37] Bruno De Man, Johan Nuyts, Patrick Dupont, Guy Marchal, and Paul Suetens. An iterative maximum-likelihood polyenergetic algorithm for CT. *IEEE Transactions on Medical Imaging*, 20(10):999–1008, 2001.

[38] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal*, 11:431–441, 1963.

[39] MathWorks. *Radon transform*, 2011. `http://www.mathworks.com/help/toolbox/images/ref/radon.html`.

[40] W. D. McDavid, R. G. Waggener, W. H. Payne, and M. J. Denis. Correction for spectral artifacts in cross-sectional reconstruction from x-rays. *Journal of Medical Physics*, 4(1):54–57, 1997.

[41] J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory, Lecture Notes in Mathematics. *Springer Verlag*, 630:105–116, 1978.

[42] Frank Natterer. *The mathematics of computerized tomography*. SIAM, Philadelphia, PA, USA, 2001.

[43] Frank Natterer and Frank Wübbeling. *Mathematical Methods in Image Reconstruction*. Medical Physics, SIAM, Philadelphia, PA, 2001.

[44] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[45] NVIDIA Corporation. *GeForce 8 and 9 Series GPU Programming Guide*, 2008. `http://developer.download.nvidia.com/GPU_Programming_Guide/GPU_Programming_Guide_G80.pdf`.

[46] James R. Parker. *Algorithms for Image Processing and Computer Vision*. John Wiley and Sons, New York, 1997.

[47] A. R. De Pierro. On the relation between the ISRA and the EM algorithm for positron emission tomography. *IEEE Transactions on Medical Imaging*, 12(2):328–333, 1993.

[48] A. R. De Pierro. A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography. *IEEE Transactions on Medical Imaging*, 14(1):132–137, 1995.

[49] Nargol Rezvani, D. A. Aruliah, John M. Boone, Michael J. Flynn, Kenneth R. Hoffmann, Kenneth R. Jackson, Douglas Moseley, Xiaochuan Pan, Jeffrey Siewerdsen, and Xiangyang Tang. OSCaR: open source cone-beam reconstructor. *American Association of Physicists in Medicine*, Technical report, submitted, 2012.

[50] Nargol Rezvani, D. A. Aruliah, John M. Boone, Michael J. Flynn, Kenneth R. Hoffmann, Kenneth R. Jackson, Douglas Moseley, Xiaochuan Pan, Jeffrey Siewerdsen, and Xiangyang Tang. An overview of OSCaR: open source cone-beam reconstructor. *American Association of Physicists in Medicine*, submitted, 2012.

[51] T. Rodet, F. Noo, and M. Defrise. The cone-beam algorithm of Feldkamp, Davis, and Kress preserves oblique line integrals. *Journal of Medical Physics*, 31(7):1972–1975, 2004.

[52] Paul Rodríguez and Brendt Wohlberg. An iteratively weighted norm algorithm for total variation regularization. *Proceedings of Fortieth Asilomar Conference on Signals, Systems, and Computers*, 43:892–896, 2006.

[53] B. Rust and D. Donnelly. The fast Fourier transform for experimentalists, part V: Filters. *Computing in Science and Engineering*, 8(1):92–95, 2005.

[54] Somesh Srivastava and Jeffrey A. Fessler. Simplified statistical image reconstruction algorithm for polyenergetic x-ray CT. *IEEE Nuclear Science Symposium Conference Record*, 3:1551–1555, 2005.

[55] Henrik Turbell. *Cone-beam reconstruction using filtered back-projection*. PhD thesis, Department of Electrical Engineering, Linköpings Universitet, 2001.

[56] C. R. Vogel and M. E. Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996.

[57] Curtis R. Vogel. *Computational Methods for Inverse Problems, Frontiers in Applied Mathematics*. SIAM, 2002.

[58] James S. Walker. *Fast Fourier Transforms*. CRC Press, 2000.

[59] S. Webb. A modified convolution reconstruction technique for divergent beams. *Physics in Medicine and Biology*, 27(3):419–423, 1982.

[60] Yuchuan Wei, Ge Wang, and Jiang Hsieh. An intuitive discussion on the ideal ramp filter in computed tomography. *Computers & Mathematics with Applications*, 49(5–6):731–740, 2005.

[61] Chye Hwang Yan, R. T. Whalen, G. S. Beaupre, S. Y. Yen, and S. Napel. Reconstruction algorithm for polychromatic CT imaging: application to beam hardening correction. *Medical Imaging, IEEE Transactions on*, 19(1):1–11, 2000.

[62] Yanbo Zhang, Xuanqin Mou, and Shaojie Tang. Beam hardening correction for fan-beam CT imaging with multiple materials. *Nuclear Science Symposium Conference Record (NSS/MIC), 2010 IEEE*, pages 3566–3570, 2010.