Iterative Correction Methods for the Beam Hardening Artifacts in Computed Tomography

Chelsea Hall
Department of Mathematics
Oregon State University

December 2013

Abstract

In this paper, we discuss three iterative correction methods proposed by G. Van Gompel et al. [15] for the beam hardening artifacts in computed tomography. We implement each method, using previous work done by M. Alarfaj [1], and provide results for comparison of the effectiveness of each method.

In loving memory of Tim Hart,
a precious friend and mentor

Acknowledgements

My deepest gratitude to my advisor, Dr. Adel Faridani, for his guidance, patience, and encouragement throughout this journey.

A special thanks to my family; to my fellow researcher and friend, Hannah Stanton; and to each of my amazing roommates for their love, support, and countless hugs along the way.

I give all the credit for everything that I have accomplished to my Lord and Savior, Jesus Christ. It is through His strength, love, and provision that I am here today. *Colossians 3:23*

Chapter 1

Introduction

1.1 X-ray Computed Tomography

X-ray computed tomography (CT) is a form of medical imaging that aims to non-invasively produce cross-sectional images of the inside of the human body using data obtained via x-rays. The fundamental problem of CT is to reconstruct the image of an object from its projections. Mathematically speaking, CT aims to reconstruct a function from its line integrals.

This problem was solved in the early 20th century by Johann Radon, who showed that a function could be reconstructed from an infinite set of its line integrals [11]. However, it wasn't until the 1960's that Allen Cormack, unfamiliar with the work of Radon, contributed the first mathematical implementation of tomographic reconstruction. Shortly thereafter, Godfrey Hounsfield developed the first CT scanner to be used commercially. Cormack and Hounsfield were awarded the Nobel Prize in Physiology or Medicine in 1979 for their respective contributions to the development of CT [3] [5].

When a CT scan is taken, the x-rays are attenuated as they pass through the body. However, they are attenuated differently depending on the material. An x-ray beam is attenuated more when it passes through a material with high density than it is when it passes through a material of low density. Thus, because the body is not homogenous, the attenuation is dependent on the path chosen. For this reason, projections must be obtained at many different angles. These projections are then used to estimate the attenuation coefficients for the different materials throughout the body. In the following pages, when we refer to the density of a material, we mean

its x-ray attenuation coefficient. Thus, with an accurate estimate of the density of the materials throughout the body, the image can be reconstructed.

There are, however, many different artifacts that can distort the reconstructed image from the true image. These artifacts have a variety of causes, including the patient, the scanner, or the physics involved. Some artifacts caused by the patient are due to metal objects present in the region being scanned, such as dental fillings or prosthetic devices, while others arise when the patient moves during the scan. Artifacts caused by the scanner could be due to the need for recalibration or even repair. Physics-based artifacts in the image have a number of different causes, one of which is due to the use of x-rays that are composed of photons of varying energy levels. This is called the beam hardening artifact. Correcting for this artifact will be the focus of the remainder of this paper. [2]

1.2 Beam Hardening

As described above, the main goal of CT is to estimate the attenuation coefficients throughout the body in order to produce a reconstruction of the image. For a monochromatic energy source, the x-ray beam is made up of photons at the same energy level. Thus, the attenuation coefficient at each point is unique and depends only on the material at that point. However, for a polychromatic energy source, the x-ray beam is made up of photons at different energy levels. As the beam passes through the body, low energy photons are absorbed more strongly than high energy photons. Thus, the attenuation of an x-ray at a point in the body depends on the path traveled as well as the energy level. Low energy x-ray beams are often called soft beams, while high energy x-ray beams are called hard beams. The composition of the polychromatic x-ray beam is more predominantly higher energy photons after it has passed through the body than it was before passing through. Thus, it is said that the x-ray beam hardens as it passes through the body. This leads to the term beam hardening. Note that beam hardening does not occur with monochromatic x-ray beams because all photons are at the same energy level.

Now, assuming an ideal monochromatic energy source, the detected intensity I_D of the x-ray beam can be expressed using Lambert-Beer's Law, giving

$$I_D = I_0 e^{-\int_L \mu(x,y)ds} \tag{1.1}$$

where I_0 is the initial intensity of the monochromatic x-ray beam, L is the line over

which the beam travels, and $\mu(x, y)$ is the linear attenuation coefficient at the point (x, y) along line L. From this equation, the monochromatic ray sum is defined to be

$$m = -ln\left(\frac{I_D}{I_0}\right) = \int_L \mu(x, y)ds \tag{1.2}$$

and is not dependent on the energy of the x-ray beam. Additionally, there is a linear relationship between μ and the monochromatic ray sum leading to an algorithm which produces a reconstructed image lacking beam hardening artifacts.

In practice, x-ray beams are not monochromatic. Thus, it is necessary to consider the case where a polychromatic energy source is used. As described above, attenuation is dependent on the energy of the photons. Thus, the detected intensity should take into account this energy dependence. Define $\tau(E)$ to be the probability density that a given photon is at energy E. Then, the detected intensity I_D is given by

$$I_D = I_0 \int_0^{E_{max}} \tau(E)e^{-\int_L \mu(x,y,E)ds}dE \tag{1.3}$$

where $\mu(x, y, E)$ is the attenuation coefficient at the point (x, y) for energy E and I_0 is the initial intensity of the x-ray beam. Defining the polychromatic ray sum in the same way that the monochromatic ray sum was defined gives

$$p = -ln\left(\frac{I_D}{I_0}\right) = -ln\left(\int_0^{E_{max}} \tau(E)e^{-\int_L \mu(x,y,E)ds}dE\right) \tag{1.4}$$

and takes into account the energy dependence of the attenuation coefficients. While the monochromatic ray sum in Eq. (1.2) had a linear relationship with μ, Eq. (1.4) yields a nonlinear relationship between the polychromatic ray sum and μ. It is this nonlinearity that causes the beam hardening artifacts in the reconstructed image when the algorithm obtained for monochromatic data is used.

1.3 The Beam Hardening Artifacts

To observe the beam hardening artifacts, we will use a 200 x 200 density phantom produced at 60 keV, as seen in Fig. 1.1. The phantom consists of one large disk, which is assigned the attenuation coefficient of a material similar to that of brain matter, and four small disks, which are assigned the attenuation coefficient of a material similar to bone. The locations of each disk are described by the coordinates

(x, y) of the center of the disk and by the radius r, with the values for each shown in Table 1.2. The linear attenuation coefficients of each material are given in Table 1.1 obtained from [9]. Unless otherwise noted, each image was produced using a grey scale window of [.17,.24].

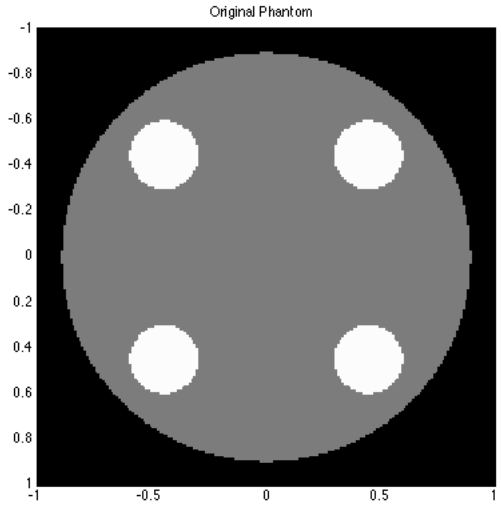
Table 1.1: Energy spectrum and linear attenuation coefficients

Energy (keV)	τ	Bone	Brain	Soft Tissue 1	Soft Tissue 2
41	0.1	0.999	0.265	0.357	0.448
52	0.3	0.595	0.226	0.272	0.3182
60	0.3	0.416	0.210	0.236	0.261
84	0.2	0.265	0.183	0.193	0.203
100	0.1	0.208	0.179	0.178	0.182

Table 1.2: Location parameters used to produce phantom

No.	x	y	r
1	0	0	0.9
2	-0.45	0.45	0.15
3	0.45	0.45	0.15
4	-0.45	-0.45	0.15
5	0.45	-0.45	0.15

Figure 1.1: Original Phantom



We assume a discretized energy spectrum consisting of energies $E_1, \ldots, E_K$ to approximate the polychromatic ray sum given in Eq. (1.4), giving

$$p^{approx} \approx -\ln\left(\sum_{k=1}^{K} \tau_k e^{-\int_L \mu(x,y,E_k)ds}\right) \tag{1.5}$$

where τ_k is approximately the probability that the given photon is at energy E_k. The discretized energy spectrum and probability density used are given in Table 1.1.

Figure 1.2 shows the original phantom in comparison to the reconstructed image using monochromatic data as explained above. Overall, the monochromatic reconstruction appears to be comparable to the original. There is some noise present; however, the discrepancies seen in the reconstructed image are not those characteristic of the beam hardening artifacts.

Figure 1.2: Reconstruction using ideal monochromatic data with grey scale [.17,.24] and picture size 200 x 200 **Top left:** Original image, **Top right:** Density profile for row 120 of original phantom, **Bottom left:** Reconstructed image using monochromatic data at 60 keV, **Bottom right:** Density profile for row 120 of monochromatic reconstruction.

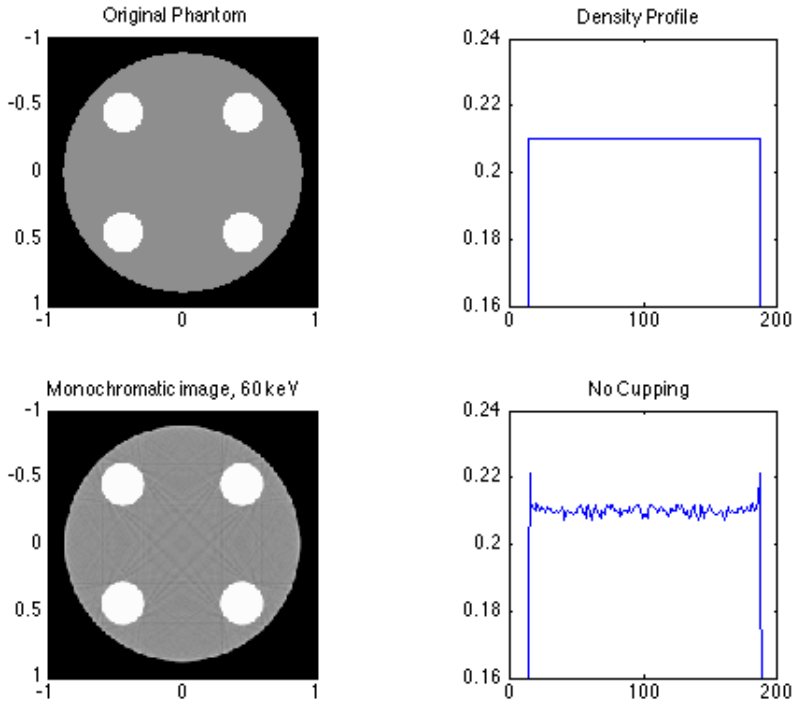
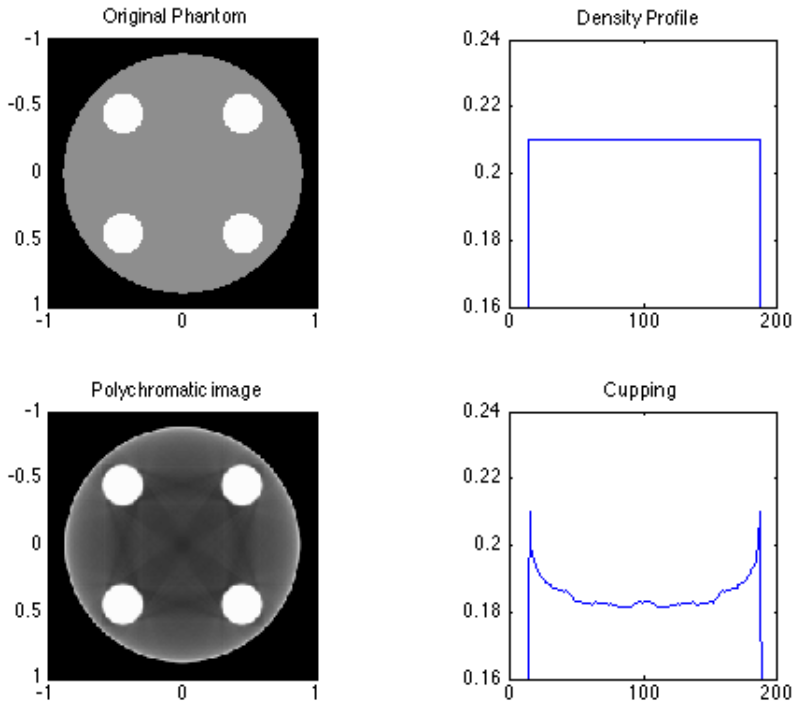


Figure 1.3 shows the original phantom in comparison to the reconstructed image using polychromatic data. Two types of artifacts are characteristic of beam hardening: the streaking artifact and the cupping artifact. The streaking artifact is characterized by dark streaks between areas of high density. The cupping artifact causes areas of constant density to appear cupped. In other words, the reconstructed density is lower in the interior of the object. Both of these artifacts are present in this reconstructed image. The dark streaks can be seen between the four disks of bone. The cupping is seen in the density profile where there are much smaller values through the middle of the object. The contrast can be seen clearly between the density profile of the original phantom compared to that of the polychromatic image below it.

Figure 1.3: Reconstruction using polychromatic data with beam hardening artifacts, grey scale [.17,.24], and picture size 200 x 200, **Top left:** Original image, **Top right:** Density profile for row 120 of original phantom, **Bottom left:** Reconstruction using polychromatic data, showing the streaking artifact, **Bottom right:** Density profile for row 120 of polychromatic reconstruction, showing the cupping artifact.



1.4 Correction of Beam Hardening Artifacts

The problem of correcting for beam hardening artifacts has been the subject of much research in the past four decades. This has lead to a variety of possible correction methods. Some of these methods include hardware filtering, linearization, dual energy, and statistical reconstruction.

Hardware filtering is a common method used to limit the energy spectrum of the x-rays at the source. As described above, low energy photons are absorbed more strongly than high energy photons. Thus, a metal plate is placed between the source and the object to absorb the low energy photons before the x-ray beam reaches the

object. While this method is effective in reducing the beam hardening artifacts, it also leads to a decrease in the signal to noise ratio [14] [15].

Linearization correction methods seek to generate monochromatic data from measured polychromatic data. For objects consisting of multiple materials, this is often done using an iterative post reconstruction approach [10] [15]. At each iteration, the sinogram is corrected using estimates for material thicknesses (the length of the intersection of the material with the x-ray path) and prior knowledge about the system, including the number of materials present in the image and how the attenuation coefficients depend on the energy level. These methods often assume a certain amount of prior knowledge which is not always readily available.

The attenuation coefficient of a material represents the effects of two absorption processes: the photo electric effect and the Compton effect [12]. Dual energy correction methods [4] model the attenuation coefficients as a linear combination of basis functions representing the respective contributions from each absorption process. One of the drawbacks of this method is that it generally requires two scans to be done with different energy spectra, increasing the dose that a patient receives.

Statistical reconstruction methods [6] [8] use a maximum likelihood algorithm to take into account the polychromatic nature of the x-ray beam. This approach assumes that the base substances are known and that the energy dependence of the attenuation coefficients can be expressed as a linear combination of the energy dependencies of the base substances. While these methods are flexible, they also have a higher cost computationally.

In this paper, we will discuss three iterative correction methods for the beam hardening artifacts presented by G. Van Gompel, K. Van Slambrouck, M. Defrise, K. Batenburg, J. de Mey, J. Sijbers, and J. Nuyts [15]. Each of these methods is based on a physical model produced using a given set of parameters. In Chapter 2, we will present each of these correction methods in greater detail. Chapter 3 describes our experiments and results while Chapter 4 provides a comparison on the effectiveness of each method.

Chapter 2

Iterative Correction Methods by Van Gompel, et al.

Three iterative methods to correct for the beam hardening artifact have been proposed in [15]. These methods are the iterative gradient based reconstruction (IGR), the iterative filtered backprojection approach (IFR), and the iterative sinogram preprocessing method (ISP). These methods assume that the number of materials in the body is known and that the energy spectrum can be represented by a small, predefined number of energy "bins". These methods can also deal with small variations in the density of the materials allowing for the reconstruction of materials that are not perfectly uniform. Additionally, they do not require prior knowledge about the properties of the materials nor are they limited to a certain number of materials in the image, unlike many other reconstruction methods.

The goal of each of these methods is to produce an image free of the cupping and streaking artifacts that are due to beam hardening. This image is produced using simulated polychromatic data obtained from a given set of estimated parameters, where the simulated data is based on a physical model. This parameter estimation is treated as an optimization problem where we aim to minimize the mean square error between the measured polychromatic data and the simulated polychromatic data obtained from the estimated parameters.

2.1 Simulated Polychromatic Data and Parameters

In order to determine the parameters to be estimated, we must first develop an algorithm for producing simulated polychromatic data. Let N be the number of materials present in the object to be reconstructed, let J be the total number of pixels in the image, and let K be the total number of energy bins. Additionally we define $\mu_n(E_k)$ to be an approximation of the linear attenuation coefficient for material n at energy E_k. In this method, we assume that each pixel contains only one material. Thus, we define $s_{n,j}$ such that $s_{n,j} = 1$ if pixel j contains material n, and $s_{n,j} = 0$ otherwise. Finally, as described above, we do not assume that each material is perfectly uniform. Therefore, we will introduce d_j to be the relative density of the material in pixel j. Now, the attenuation at pixel j for energy E_k can be expressed by

$$A_{j,k} = d_j \sum_{n=1}^{N} \mu_n(E_k) s_{n,j} \tag{2.1}$$

Then, the line integral along line i at energy E_k of the discretized image can be expressed as

$$\begin{aligned} L_{i,k} &= \sum_{j=1}^{J} l_{i,j} A_{j,k} \\ &= \sum_{n=1}^{N} \mu_n(E_k) \sum_{j=1}^{J} l_{i,j} d_j s_{n,j} \end{aligned} \tag{2.2}$$

where $l_{i,j}$ is the length of the intersection of line i with pixel j. Thus, the simulated polychromatic data along line i can be expressed by

$$p_i^{sim} = \ln \left(\sum_{k=1}^{K} \tau_k e^{-\sum_{n=1}^{N} \mu_n(E_k) \sum_{j=1}^{J} l_{i,j} d_j s_{n,j}} \right) \tag{2.3}$$

From this expression of the simulated data, we are able to determine our parameters to be the attenuation coefficients μ, the fractional intensity τ, the relative density d, and the segmentation s.

2.2 Cost Function

The goal of the iterative correction methods is to produce simulated polychromatic data that closely matches the measured data. In other words, the methods iteratively minimize a cost function Φ to produce an image free of the beam hardening artifacts. Let M be the total number of x-ray projection lines. Then, we define

$$\Phi(\mu, \tau, d, s) = \frac{1}{M} \sum_{i=1}^{M} (p_i^{meas} - p_i^{sim})^2 \tag{2.4}$$

where p_i^{meas} is the measured polychromatic data and p_i^{sim} is the simulated polychromatic data. The measured data along line i is defined to be

$$p_i^{meas} = \ln\left(\frac{I_{D,i}}{I_0}\right) \tag{2.5}$$

and the simulated data is given in Eq. (2.3). Thus, we can rewrite the cost function given in Eq. (2.4) as

$$\Phi(\mu, \tau, d, s) = \frac{1}{M} \sum_{i=1}^{M} \left(\ln\left(\frac{I_{D,i}}{I_0}\right) - \ln\left(\sum_{k=1}^{K} \tau_k e^{-\sum_{n=1}^{N} \mu_n(E_k) \sum_{j=1}^{J} l_{i,j} d_j s_{n,j}} \right) \right)^2 \tag{2.6}$$

When computing the simulated polychromatic data in our implementation, we use MATLAB's built-in Radon command to approximate $L_{i,k}$, the expression for the line integral, given in Eq.(2.2). Also note that the cost function Φ does not have a unique minimum. For example,

$$\Phi(\mu, \tau, d, s) = \Phi(\mu/2, \tau, 2d, s) \tag{2.7}$$

Thus, it is important to initialize the following procedure with good estimates of the parameters.

2.3 Iterative Gradient based Reconstruction (IGR)

As stated previously, the iterative gradient based reconstruction (IGR) method aims to iteratively minimize the cost function given in Eq. (2.6). To do this, each step aims to minimize the cost function for a specific parameter while holding the other parameters constant rather than attempting to minimize for all parameters at once. The

IGR method begins with the user selecting the number of materials N and number energy bins K, and by performing an initial reconstruction of the measured polychromatic data using the filtered back projection algorithm [16], $R^0 = FBP(p_i^{meas})$, in order to obtain good estimates for the parameters. Then, the following occurs at each iteration, w:

2.3.1 Update the segmentation

To estimate the parameter $s_{n,j}$, we split the current image, R^{w-1}, into N segments via thresholding so as to minimize the cost function. However, the gradient of the cost function with respect to segmentation is very often near zero. We only see a considerable change in the cost function when the threshold value is close to the density of one of the materials. When this happens, changing the threshold slightly results in a change in the segmentation. When the threshold value is not close to the density of one of the materials, the segmentation stays essentially the same when the threshold value is changed slightly. Because of this, it proved difficult to find threshold values that minimized the cost function using MATLAB's standard optimization tools. Thus, we implemented an adhoc method developed by A. Faridani to find initial threshold values for which the gradient of the cost function is not zero.

In this method, we define a function $F(x)$ to be the number of pixels in the current image with densities less than or equal to x. As x increases, we see that $F(x)$ also increases because there are more pixels less than or equal to x. Now, $F(x)$ changes most drastically when x is near the density of one of the materials. Thus, we approximate $F'(x)$ and then search for peaks in the data. In other words, we find densities at which the local maxima of $F'(x)$ occur and choose those to be our initial threshold values. We then use a steepest descent method to find thresholds that minimize the cost function. More specifically, for this we approximate the gradient using finite differences to find the direction of steepest descent. We then estimate a step size relative to the threshold values obtained. At this point, we update the thresholds and reevaluate the cost function. We continue to do this until the cost function stops decreasing, giving us the new estimate for our thresholds and segmentation.

2.3.2 Update the relative density

We initialize the relative density, d_j, by one in the first iteration for all pixels in the image. For each iteration $w > 1$,we use the updated value for s and estimate the

relative density parameter by minimizing the cost function [Eq. (2.6)]. To do this, G. Van Gompel et al. [15] develop a gradient descent algorithm by seeking a surrogate function $\Phi_A(\mathbf{d}, \mathbf{d}^n)$ of the cost function given in Eq. (2.6), which satisfies the conditions $\Phi_A(\mathbf{d}, \mathbf{d}^n) \geq \Phi(\mathbf{d})$ and $\Phi_A(\mathbf{d}^n, \mathbf{d}^n) = \Phi(\mathbf{d}^n)$ [7] [13]. All other parameters are held constant during the relative density update. Thus, they are not included here in the notation. The goal is to obtain a function that is more easily minimized than the original. To do this, a surrogate function is built for the approximation of Φ given by a quadratic Taylor expansion:

$$\Phi(\mathbf{d}) \simeq \Phi(\mathbf{d}^n) + \sum_{j=1}^{J} \left.\frac{\partial \Phi}{\partial d_j}\right|_{d^n} (d_j - d_j^n) + \frac{1}{2} \sum_{j=1}^{J} \sum_{j=1}^{J} \left.\frac{\partial^2 \Phi}{\partial d_j \partial d_h}\right|_{d^n} (d_j - d_j^n)(d_h - d_h^n) \quad (2.8)$$

Now, the first derivative of Φ is given by:

$$\begin{aligned} \frac{\partial \Phi(\mathbf{d})}{\partial d_j} =& \frac{\partial}{\partial d_j} \left(\frac{1}{M} \sum_{i=1}^{M} \left(\ln \left(\frac{I_{D,i}}{I_0} \right) \right.\right. \\ & \left.\left. - \ln \left(\sum_{k=1}^{K} \tau_k e^{-\sum_{n=1}^{N} \mu_n(E_k) \sum_{j=1}^{J} l_{i,j} d_j s_{n,j}} \right) \right)^2 \right) \end{aligned} \quad (2.9)$$

Defining

$$P_{i,k} = \tau_k e^{-\sum_{n=1}^{N} \mu_n(E_k) \sum_{j=1}^{J} l_{i,j} d_j s_{n,j}} \quad (2.10)$$

$$P_i = \sum_{k=1}^{K} P_{i,k} \quad (2.11)$$

allows Eq. (2.9) to be rewritten as

$$
\begin{aligned}
\frac{\partial \Phi(\mathbf{d})}{\partial d_j} &= \frac{\partial}{\partial d_j}\left(\frac{1}{M}\sum_{i=1}^{M}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln\left(P_i\right)\right)^2\right) \\
&= \frac{2}{M}\sum_{i=1}^{M}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\left(\sum_{k=1}^{K}\frac{P_{i,k}}{P_i}\sum_{n=1}^{N}\mu_n(E_k)l_{i,j}s_{n,j}\right) \\
&= \frac{2}{M}\sum_{k=1}^{K}\sum_{i=1}^{M}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\left(\frac{P_{i,k}}{P_i}\sum_{n=1}^{N}\mu_n(E_k)l_{i,j}s_{n,j}\right) \\
&= \frac{2}{M}\sum_{k=1}^{K}\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\sum_{i=1}^{M}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{P_{i,k}}{P_i}l_{i,j} \\
&= \frac{2}{M}\sum_{k=1}^{K}\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\sum_{i=1}^{M}l_{i,j}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{P_{i,k}}{P_i}
\end{aligned}
\tag{2.12}
$$

The second derivative in Eq. (2.8) is simplified by assuming that $\left(ln\left(\frac{I_{D,i}}{I_0}\right)-ln(P_i)\right)$ can be neglected when close enough to the minimum, which leaves only the dominant term. The second derivative is given by

$$
\begin{aligned}
\frac{\partial^2 \Phi(\mathbf{d})}{\partial d_h \partial d_j} &= \frac{\partial}{\partial d_h}\left(\frac{2}{M}\sum_{k=1}^{K}\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\sum_{i=1}^{M}l_{i,j}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{P_{i,k}}{P_i}\right) \\
&= \frac{2}{M}\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right]\left[\sum_{i=1}^{M}l_{i,j}\left(\frac{\partial}{\partial d_h}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{P_{i,k}}{P_i}\right.\right. \\
&\quad \left.\left.+\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{\partial}{\partial d_h}\left(\frac{P_{i,k}}{P_i}\right)\right)\right]
\end{aligned}
\tag{2.13}
$$

Applying the approximation described allows Eq. (2.13) to be rewritten as

$$\begin{aligned}
\frac{\partial^2 \Phi(\mathbf{d})}{\partial d_h \partial d_j} &\simeq H_{j,h}(\mathbf{d}) \\
&\doteq \frac{2}{M} \sum_{k=1}^{K} \left[\sum_{n=1}^{N} \mu_n(E_k) s_{n,j} \right] \left[\sum_{i=1}^{M} l_{i,j} \left(\frac{\partial}{\partial d_h} \left(\ln \left(\frac{I_{D,i}}{I_0} \right) - \ln(P_i) \right) \right) \frac{P_{i,k}}{P_i} \right] \\
&= \frac{2}{M} \sum_{k=1}^{K} \left[\sum_{n=1}^{N} \mu_n(E_k) s_{n,j} \right] \left[\sum_{i=1}^{M} l_{i,j} \frac{P_{i,k}}{P_i} \left(\sum_{k'=1}^{K} \sum_{n'=1}^{N} \mu_{n'}(E_{k'}) s_{n',h} l_{i,h} \frac{P_{i,k'}}{P_i} \right) \right] \\
&= \frac{2}{M} \sum_{k=1}^{K} \left[\sum_{n=1}^{N} \mu_n(E_k) s_{n,j} \right] \\
&\quad \cdot \sum_{i=1}^{M} l_{i,j} l_{i,h} \frac{P_{i,k}}{P_i^2} \left(\sum_{k'=1}^{K} P_{i,k'} \sum_{n'=1}^{N} \mu_{n'}(E_{k'}) s_{n',h} \right)
\end{aligned} \tag{2.14}$$

Now, the right hand side of Eq. (2.14) is always positive and

$$2(d_j - d_j^n)(d_h - d_h^n) \leq (d_j - d_j^n)^2 + (d_h - d_h^n)^2 \tag{2.15}$$

Thus, it follows that

$$\sum_{j=1}^{J} \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)(d_j - d_j^n)(d_h - d_h^n) \leq \sum_{j=1}^{J} \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)(d_j - d_j^n)^2 \tag{2.16}$$

Applying this inequality to Eq. (2.8), the surrogate function Φ_A is defined to be

$$\begin{aligned}
\Phi_A(\mathbf{d}, \mathbf{d}^n) = \Phi(\mathbf{d}) &+ \sum_{j=1}^{J} \left. \frac{\partial \Phi}{\partial d_j} \right|_{\mathbf{d}^n} (d_j - d_j^n) \\
&+ \frac{1}{2} \sum_{j=1}^{J} \sum_{h=1}^{J} \left. \frac{\partial^2 \Phi}{\partial d_j \partial d_h} \right|_{\mathbf{d}^n} (d_j - d_j^n)^2
\end{aligned} \tag{2.17}$$

which satisfies the conditions $\Phi_A(\mathbf{d}, \mathbf{d}^n) \geq \Phi(\mathbf{d})$ and $\Phi_A(\mathbf{d}^n, \mathbf{d}^n) = \Phi(\mathbf{d}^n)$. With the new surrogate function, minimization is reduced to setting the derivative of Φ_A with respect to d_j equal to zero and solving for d_j. Thus, we have

$$
\begin{aligned}
0 &= \frac{\partial}{\partial d_j}\Phi_A \\
&= \left.\frac{\partial \Phi}{\partial d_j}\right|_{\mathbf{d}^n} + \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)(d_j - d_j^n) \\
&= \left.\frac{\partial \Phi}{\partial d_j}\right|_{\mathbf{d}^n} + \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)\cdot d_j - \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)\cdot d_j^n
\end{aligned} \tag{2.18}
$$

Rearranging the terms in Eq. (2.18), we obtain

$$
\begin{aligned}
\sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)\cdot d_j &= \sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)\cdot d_j^n - \left.\frac{\partial \Phi}{\partial d_j}\right|_{\mathbf{d}^n} \\
d_j &= d_j^n - \frac{\left.\frac{\partial \Phi}{\partial d_j}\right|_{\mathbf{d}^n}}{\sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)},
\end{aligned} \tag{2.19}
$$

corresponding to a gradient descent with diagonal preconditioner $\frac{1}{\sum_{h=1}^{J} H_{j,h}(\mathbf{d}^n)}$. With a nonnegativity constraint added, this leads to a relative density update of

$$
\begin{aligned}
d_j^{n+1} = \Bigg[& d_j^n \\
& - \frac{\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right]\left[\sum_{i=1}^{M} l_{i,j}\left(\ln\left(\frac{I_{D,i}}{I_0}\right) - \ln(P_i)\right)\frac{P_{i,k}}{P_i}\right]}{\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right]\sum_{i=1}^{M} l_{i,j}\frac{P_{i,k}}{P_i^2}\sum_{k'=1}^{K} P_{i,k'}\sum_{n'=1}^{N}\mu_{n'}(E_{k'})\sum_{h=1}^{J} l_{i,h}s_{n',h}}\Bigg]_+
\end{aligned} \tag{2.20}
$$

where $[x]_+ = x$ if $x \geq 0$ and $[x]_+ = 0$ if $x < 0$. However, the second derivative in Eq. (2.14) uses one backprojection $\left(\text{represented by } \sum_i l_{i,j}\right)$ for each energy bin k. To reduce the number of backprojections required, each term of the second derivative is multiplied by $\left(\frac{P_i}{P_{i,k}}\right) > 1$. Thus, the second derivative becomes

$$
\begin{aligned}
H_{j,h}(\mathbf{d}) &= \frac{2}{M}\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right] \\
&\quad \cdot \sum_{i=1}^{M} l_{i,j}l_{i,h}\frac{P_{i,k}}{P_i^2}\left(\sum_{k'=1}^{K}P_{i,k'}\sum_{n'=1}^{N}\mu_{n'}(E_{k'})s_{n',h}\right)\left(\frac{P_i}{P_{i,k}}\right) \\
&= \frac{2}{M}\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right] \\
&\quad \cdot \sum_{i=1}^{M} l_{i,j}l_{i,h}\frac{1}{P_i}\left(\sum_{k'=1}^{K}P_{i,k'}\sum_{n'=1}^{N}\mu_{n'}(E_{k'})s_{n',h}\right) \qquad (2.21)
\end{aligned}
$$

and now contains only one backprojection. Because this approximation decreases the step size, it does not lead to instability. The new relative density update with this approximation now becomes

$$
\begin{aligned}
d_j^{n+1} = \Bigg[& d_j^n \\
& - \frac{\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right]\left[\sum_{i=1}^{M} l_{i,j}\left(\ln\left(\frac{I_{D,i}}{I_0}\right)-\ln(P_i)\right)\frac{P_{i,k}}{P_i}\right]}{\sum_{k=1}^{K}\left[\sum_{n=1}^{N}\mu_n(E_k)s_{n,j}\right]\sum_{i=1}^{M} l_{i,j}\frac{1}{P_i}\sum_{k'=1}^{K}P_{i,k'}\sum_{n'=1}^{N}\mu_{n'}(E_{k'})\sum_{h=1}^{J}l_{i,h}s_{n',h}}\Bigg]_+
\end{aligned} \qquad (2.22)
$$

with $[x]_+ = x$ if $x \geq 0$ and $[x]_+ = 0$ if $x < 0$.

We implement the relative density update described in Eq.(2.20), which we will call the Full IGR method (without the approximation), as well as the update described in Eq. (2.22), which we will call the Approximated IGR method.

2.3.3 Update the attenuation coefficients and fractional intensity

Using the updated values for s and d, the parameters μ and τ are estimated by minimizing the cost function. However, in our implementation, we use the known values of for these parameters as shown in Table (1.1).

2.3.4 Update the image

Estimated values have now been found for all of the parameters. Thus, the image is updated for segmentation at the next iteration by

$$R^w = d_j^w \sum_{n=1}^{N} \mu_{IGR,n}^w(E_k) s_{n,j}^w \tag{2.23}$$

where $\mu_{IGR,n}^w$ is the median of $\{\mu_n^w(E_1), \ldots, \mu_n^w(E_k)\}$. The stopping criteria suggested by G. Van Gompel, et al is

$$\frac{\epsilon^w + \epsilon^{w-1}}{\epsilon^{w-2} + \epsilon^{w-3}} > t \tag{2.24}$$

with $0 < t < 1$ a threshold value and where $\epsilon^w = \Phi(\mu^w, \tau^w, d^w, s^w)$ is defined to be the polychromatic error of iteration w.

2.4 Iterative Filtered Backprojection (IFR)

The iterative filtered backprojection (IFR) method aims to accelerate the relative density update step (step 2) of the IGR method. Thus, the IFR method is the same as the IGR method at all steps except step 2, the relative density update step. Similarly to the IGR method, the IFR method begins with the user selecting the number of materials N and number energy bins K, and by performing an initial reconstruction of the measured polychromatic data using the filtered back projection algorithm, $R^0 = FBP(p_i^{mesa})$, in order to obtain good estimates for the parameters. Then, the following occurs at each iteration, w:

2.4.1 Update the segmentation

We follow the same procedure as described in the IGR method to update the segmentation parameter $s_{n,j}$.

2.4.2 Update the relative density

We initialize the relative density, d_j, by one in the first iteration for all pixels in the image. Now, G. Van Gompel et al. found that attempting to estimate d_j by minimizing the cost function was very computationally complex. Thus, we implement their suggested update. For each iteration $w > 1$, we update the relative density with

$$d^w = d^{w-1} + \omega^w \cdot FBP(p^{meas} - p^{sim}(\mu^{w-1}, \tau^{w-1}, d^{w-1}, s^w)) \tag{2.25}$$

where the dot denotes element wise multiplication and ω^w is a relaxation factor with diagonal elements

$$\omega_j^w = \frac{1}{\sum_{n=1}^{N} \mu_{IFR,n}^{w-1} s_{n,j}^w} \tag{2.26}$$

where $\mu_{IFR,n}^w = max\{\mu_n^w(E_1), \ldots, \mu_n^w(E_k)\}$. Here, p^{sim} is computed using Eq. (2.3). The drawback here is that the cost function is not guaranteed to decrease.

2.4.3 Update the attenuation coefficients and fractional intensity

Using the updated values for s and d, the parameters μ and τ are estimated by minimizing the cost function. However, in our implementation, we use the known values of for these parameters as shown in Table (1.1).

2.4.4 Update the image

We follow the same procedure as described in the IGR method to update the image R^w.

2.5 Iterative Sinogram Preprocessing (ISP) method

In the iterative sinogram preprocessing (ISP) method, the relative density term d is omitted in an effort to eliminate negative effects on the conditioning of the optimization problem. However, the method still allows for the correction of errors in the segmentation. As with the previous two methods, the ISP method begins with an initial reconstruction of the measured polychromatic data, $R^0 = FBP(p_i^{mesa})$. The following occurs at each iteration, w:

2.5.1 Update the segmentation

We follow the same procedure as described in the IGR method to update the segmentation parameter $s_{n,j}$.

2.5.2 Update the attenuation coefficients and the fractional intensity

Using the updated values for s and fixing $d = 1$, the parameters μ and τ are estimated by minimizing the cost function. However, in our implementation, we use the known values of for these parameters as shown in Table (1.1).

2.5.3 Reference attenuation and mono- and polychromatic simulation

First, the reference attenuation coefficients $\mu^w_{ISP,n}$ are calculated (see description below). Then, the monochromatic and polychromatic simulations are calculated by

$$m_i^{sim,w} = \sum_{n=1}^{N} \mu^w_{ISP,n} \sum_{j=1}^{J} l_{i,j} s^w_{n,j} \tag{2.27}$$

and

$$p_i^{sim,w} = \ln \left(\sum_{k=1}^{K} \tau_k^w e^{-\sum_{n=1}^{N} \mu_n^w(E_k) \sum_{j=1}^{J} l_{i,j} s^w_{n,j}} \right). \tag{2.28}$$

2.5.4 Sinogram correction and image update

The corrected, monochromatized sinogram $m^{corr,w}$ is calculated with

$$m^{corr,w} = p^{meas} + \left(m^{sim,w} - p^{sim,w} \right) \tag{2.29}$$

For the next iteration, the image is given by $R^w = FBP(m^{core,w})$.

Reference attenuation coefficients

The reference attenuation coefficients are computed in order to correct for inaccuracies in the segmentation. The updated sinogram in Eq. (2.29) consists of the original measured sinogram as well the correction term produced from the estimated parameters. Now, if two different low-density materials were classified as the same material during segmentation, the information distinguishing the two materials from each other is in the measured sinogram, not the correction term. Thus, we would like to to select reference attenuation coefficients that minimize the magnitude of the correction term in Eq. (2.29) so that the information present in the measured

sinogram is not obscured. In other words, we would like the reference attenuation coefficients to minimize

$$\begin{aligned}\Psi(\mu^{\mathbf{w}}_{\mathbf{ISP}}) &= \sum_{i=1}^{M}\left(m_i^{sim,w} - p_i^{sim,w}\right)^2 \\ &= \sum_{i=1}^{M}\left(\sum_{n=1}^{N}\mu^w_{ISP,n}t^w_{n,i} + \ln\left(\sum_{k=1}^{K}\tau_k^w e^{-\sum_{n=1}^{N}\mu_n^w(E_k)t^w_{n,i}}\right)\right)\end{aligned}$$

where

$$t^w_{n,i} = \sum_{j=1}^{J} l_{i,j}s^w_{n,j}.$$

The solution to this minimization problem gives

$$\mu^{\mathbf{w}}_{\mathbf{ISP}} = \mathbf{B}^{w+}\mathbf{v}$$

where $\mathbf{B}^{w+}$ is the psuedoinverse of the matrix $\mathbf{B}^w$ with elements

$$b^w_{n,n'} = \sum_{i=1}^{M} t^w_{n,i}t^w_{n',i}$$

for $n, n' = 1, \ldots, N$, and the vector $\mathbf{v}$ has elements

$$v^w_n = -\sum_{i=1}^{M} t^w_{n,i}\ln\left(\sum_{k=1}^{K}\tau_k^w e^{-\sum_{n=1}^{N}\mu_n^w(E_k)t^w_{n,i}}\right)$$

for $n = 1, \ldots, N$.

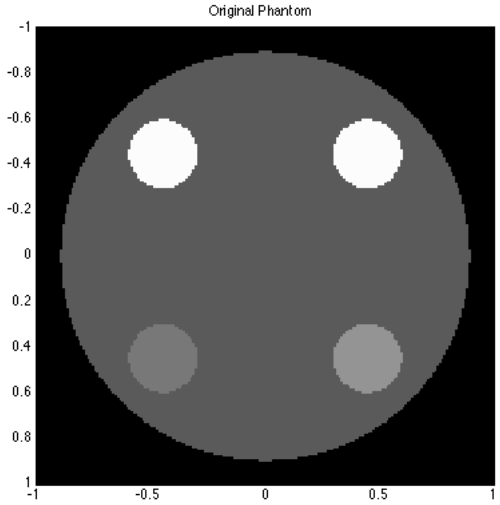
Chapter 3

Experiments and Results

3.1 Modified Phantom

To test the effectiveness of each of the methods for reducing beam hardening artifacts, we use a slightly modified phantom from the one previously discussed. Because this method allows for many materials, we modify the phantom in Fig. 1.1 to include five materials rather than three. Thus, we now have a 200 x 200 density phantom produced at 60 keV that consists of one large disk, which is assigned the attenuation coefficient of a material similar to that of brain matter; two small upper disks, which are assigned the attenuation coefficients of bone; and two small lower disks, which are assigned the attenuation coefficients of two low contrast material, soft tissue 1 and soft tissue 2, respectively. This new phantom is seen in Fig. 3.1. The locations of each disk are the same as previously described in Table 1.2. The linear attenuation coefficients of all materials are given in Table 1.1.

Figure 3.1: Modified Phantom, with grey scale [.13,.35]



3.2 Comparison of Full IGR method with Approximated IGR method

As described in Section 2.3.2, we implement two versions of the IGR method: the Approximated IGR method [Eq. (2.22)] and the Full IGR method (without the approximation) [Eq. (2.20)]. Here, the Approximated IGR method attempts to limit the number of backprojections required to calculate the density, leading to a faster update. We use MATLAB's built in iradon command to calculate each backprojection. To test the effectiveness of this approximation, we implemented both versions of the IGR method on the modified phantom described with the number of iterations set at 10.

For the Full IGR method, the iradon command was called 101 times, accounting for approximately 9.5 % of the total implementation time. In contrast, 60 % of the total time was required for the update the segmentation. For the Approximated IGR method, the iradon command was called 61 times, accounting for approximately 5.9 % of the total implementation time. Comparatively, the segmentation update required 63.5 % of the total time. However, both methods consistently required the same amount amount of time to complete 10 iterations. Therefore, while we see that the backprojections required a smaller percentage of the total time for the

Approximated IGR method, this does not correspond to a decrease in the total implementation time. Additionally, G. Van Gompel et al. [15] developed the IFR method to reduce the computational complexity of the relative density update. We see here that it is the optimization of the segmentation that requires the majority of the total implementation time, not the density update. This could be due to the particular segmentation algorithm that we implemented. Thus, an improved segmentation algorithm may lead to a decrease in the time required for the optimization of the segmentation.

3.2.1 Full IGR Method (without the approximation)

Figure 3.2 shows the updated image after 20 iterations of the Full IGR method. Overall, both the cupping and streaking artifacts are significantly reduced in the reconstructed image. However, there are significantly higher values at the edge of the object, as seen in the density profile. There is also a thicker ring around the edge of the object of material that has been segmented incorrectly. This could be due to the presence of the cupping artifact in the initial reconstruction. Additionally, from the density profile, we see that this method is able to make a slight distinction between the two low density materials. While these low density regions were segmented as the same material, the relative density parameter is able to make this small distinction.

Figure 3.3 shows the corresponding values of the cost function for 40 iterations. After approximately 20 iterations, the cost function shows only a very small decrease. Thus, we see that the cost function has converged to a local minimum. However, we see a significant decrease between iteration 15 and iteration 20 due to changes in the segmentation. Unfortunately, these changes in segmentation do not improve the image, i.e. the updated image looks less like the original image. More specifically, the update changes certain pixels along the edge of the object that were segmented as soft tissue 1 to soft tissue 2, a material of higher density. In the original image, these pixels are occupied by brain material, which has a lower density than both soft tissues.

Figure 3.2: Full IGR method (without approx.) after 20 iteration, **Top left:** Reconstruction of measured polychromatic data, **Top right:** Density profile for row 135 of polychromatic reconstruction, **Bottom left:** Segmented image, **Bottom right:** Reconstructed image using Full IGR method at 20 iterations.

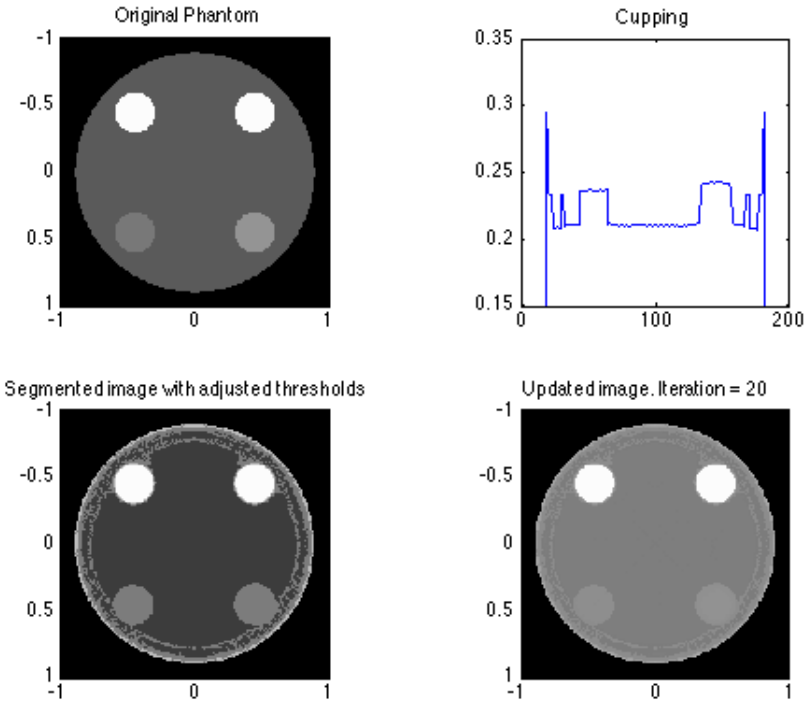
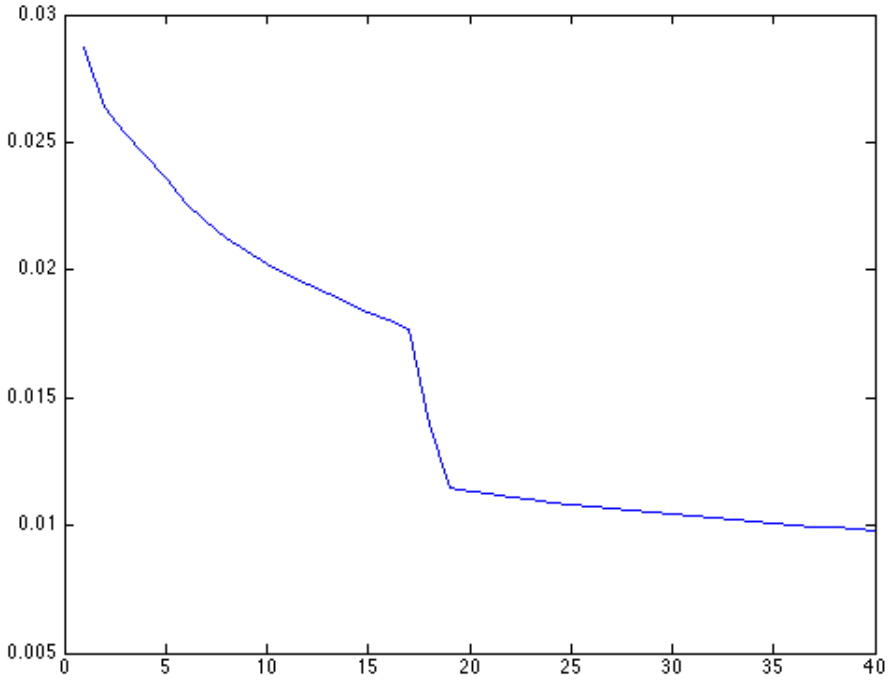


Figure 3.3: Full IGR Cost Function values vs. Iteration



3.2.2 Approximated IGR Method

Figure 3.4 shows the updated image from the Approximated IGR method after 94 iterations. The results we see here are very similar to those that we saw with the Full IGR method after 20 iterations. There is an overall reduction in the streaking and cupping artifacts. However, we still see higher values at the edge of the object, as seen in the density profile, as well as a thicker ring of material along the edge that has been incorrectly segmented as soft tissue rather than brain. This method also makes a slight distinction between the two low density materials by means of the relative density parameter.

Figure 3.4: Approximated IGR method after 94 iteration, **Top left:** Reconstruction of measured polychromatic data, **Top right:** Density profile for row 135 of polychromatic reconstruction, **Bottom left:** Segmented image, **Bottom right:** Reconstructed image using Approximated IGR method at 94 iterations.

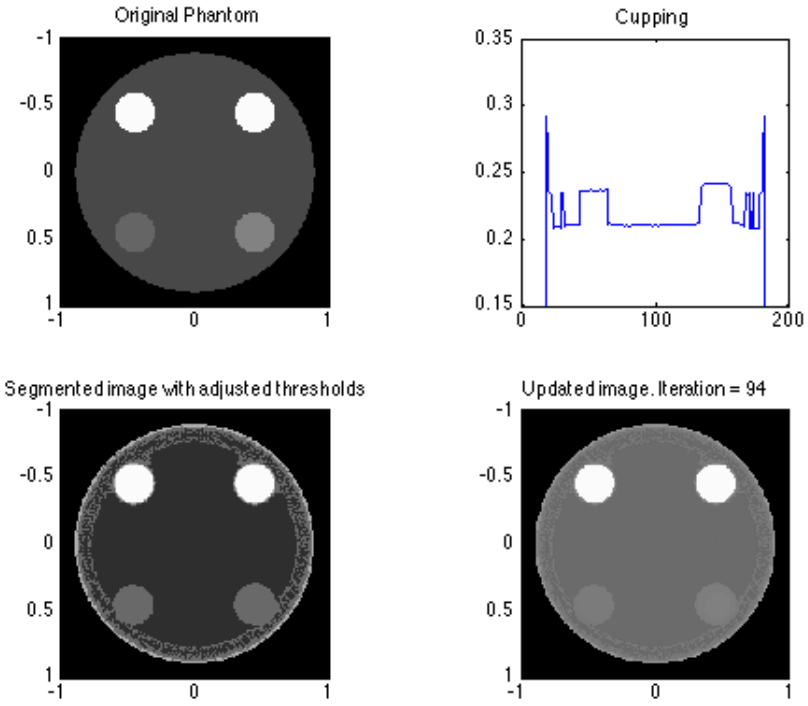
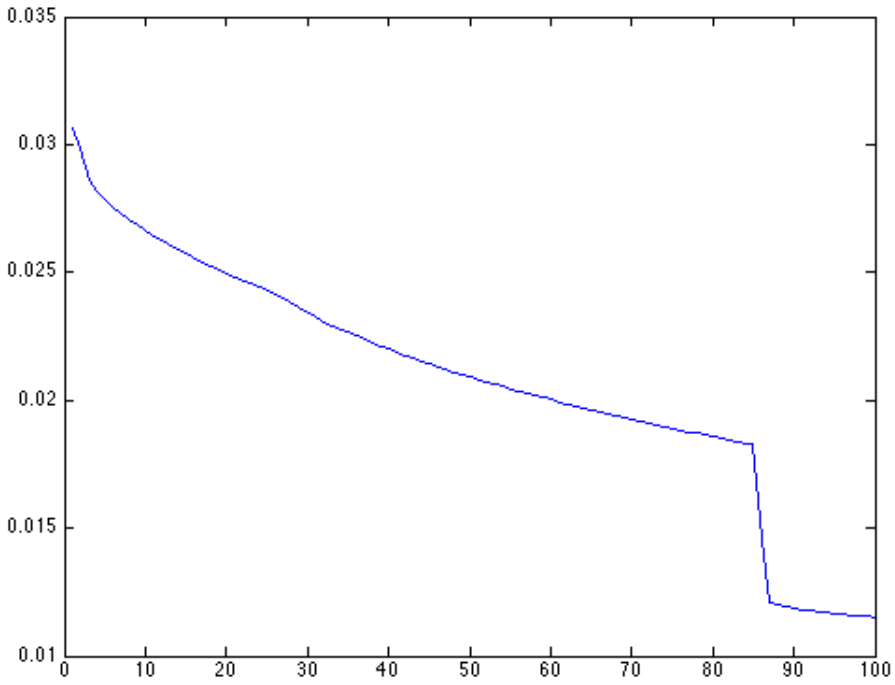


Figure 3.5 shows the values of the cost function at each iteration of the Approximated IGR method for 100 iterations. We see the cost function continues to decrease steadily until its drop significantly around iteration 85. After this, we see only a very small decrease. Again, our cost function has converged to a local minimum.

As stated previously, we obtained very similar results from the Approximated IGR method after 94 iterations as we did with the Full IGR method after just 20 iterations. As described by G. Van Gompel et al, the approximation we make to obtain the density update defined in Eq. (2.22) does not lead to instability because it decreases the step size. With a decreased step size, we would expect the Approximated IGR method to require more iterations to reach a minimum than the Full IGR method without the approximation, which is in fact what we see. Additionally, the thicker

ring of incorrectly segmented data around the edge of the object in both versions of the IGR method could be due to the presence of the cupping artifact in the initial reconstruction from the polychromatic data. It may be possible to correct for this ring with an improved algorithm to optimize the segmentation parameter.

Figure 3.5: Approximated IGR method Cost Function values vs. Iteration



3.3 IFR Method

All results were obtained using code developed by M. Alarfaj [1]. Figure 3.6 shows the updated images after 4 iterations of the IFR method compared to the original phantom. While there are still very high values around the edges of the object, as seen in the density profile, overall the cupping is not present in the reconstructed image. Here, we also see that this method was able to distinguish between the two low contrast materials present. Additionally, the dark streaks between the regions

of high density have been completely removed.

Figure 3.6: IFR results after 4 iterations, **Top left:** Reconstruction of measured polychromatic data, **Top right:** Density profile for row 135 of polychromatic reconstruction, **Bottom left:** Segmented image, **Bottom right:** Reconstructed image using IFR method at 4 iterations.

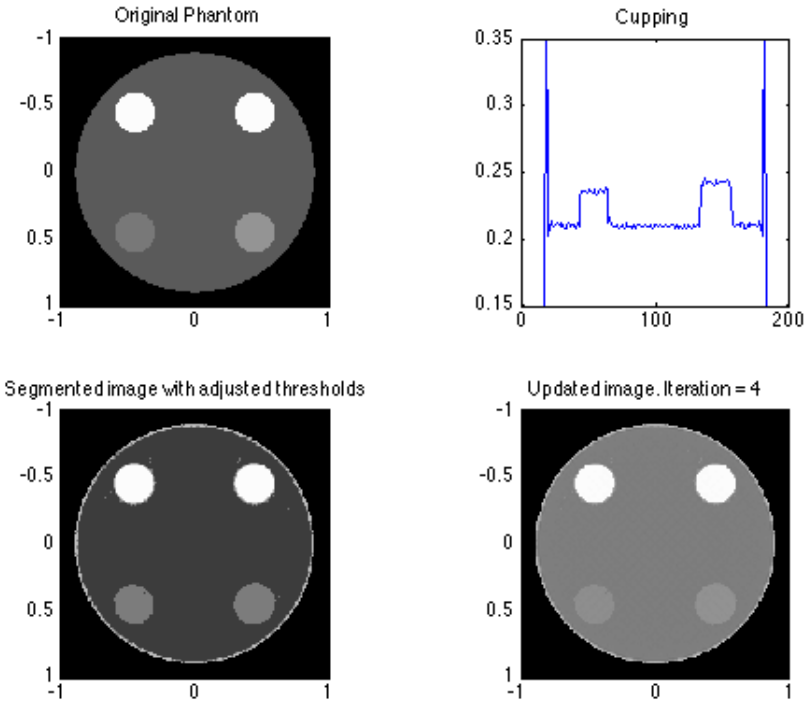
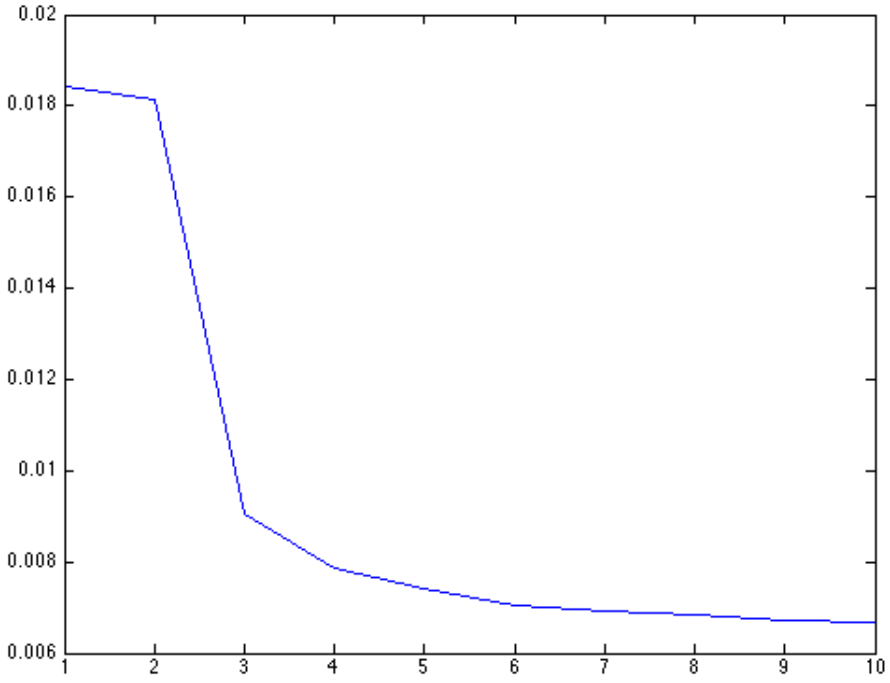


Figure 3.7 shows the values of the cost function at each iteration. We see here that the cost function is reduced at each iteration; however, the majority of the reduction occurs in the first four iterations. After that, the reduction in the cost function is minimal. Thus, we see again that the cost function has converged to a local minimum.

Figure 3.7: IFR Cost Function values vs. Iteration



3.4 ISP Method

Results were obtained using code developed by M. Alarfaj [1]. Figure 3.8 shows the updated image after 4 iterations of the ISP method. The streaking artifact is no longer present in the reconstructed image and the cupping artifact is substantially reduced. However, we see a thick ring of material around the edge of the object that is incorrectly segmented as soft material instead brain, similar to the results of the IGR method. Despite the absence of the relative density parameter, the ISP method is able to make a slight distinction between the two low-density materials, as seen in the density profile. Unlike with previous methods, the image produced with the ISP method has a lower resolution with the edges of the materials appearing less sharp.

Figure 3.8: ISP results after 4 iterations, **Top left:** Reconstruction of measured polychromatic data, **Top right:** Density profile for row 135 of polychromatic reconstruction, **Bottom left:** Segmented image, **Bottom right:** Reconstructed image using ISP method at 4 iterations.

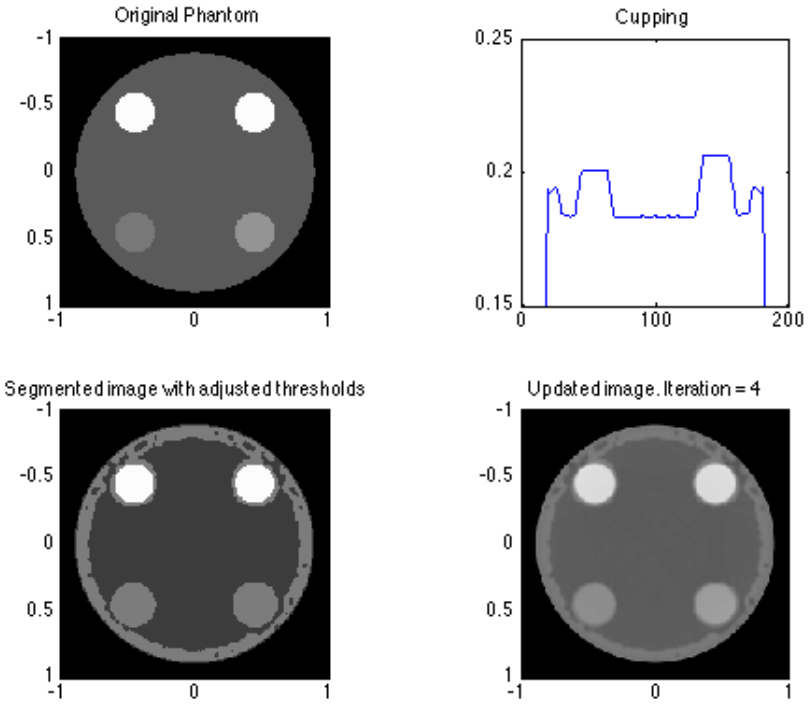


Figure 3.9 shows the values of the cost function at each iteration of the ISP method. This method is not guaranteed to decrease the cost function, which is seen here. As the cost function increases, the quality of the reconstructed image decreases, as seen in Figure 3.10. In other words, the reconstructed image looks less like the original image as we take more iterations. We do not see the cost function converge to a minimum in this case.

Figure 3.9: ISP Cost Function values vs. Iteration

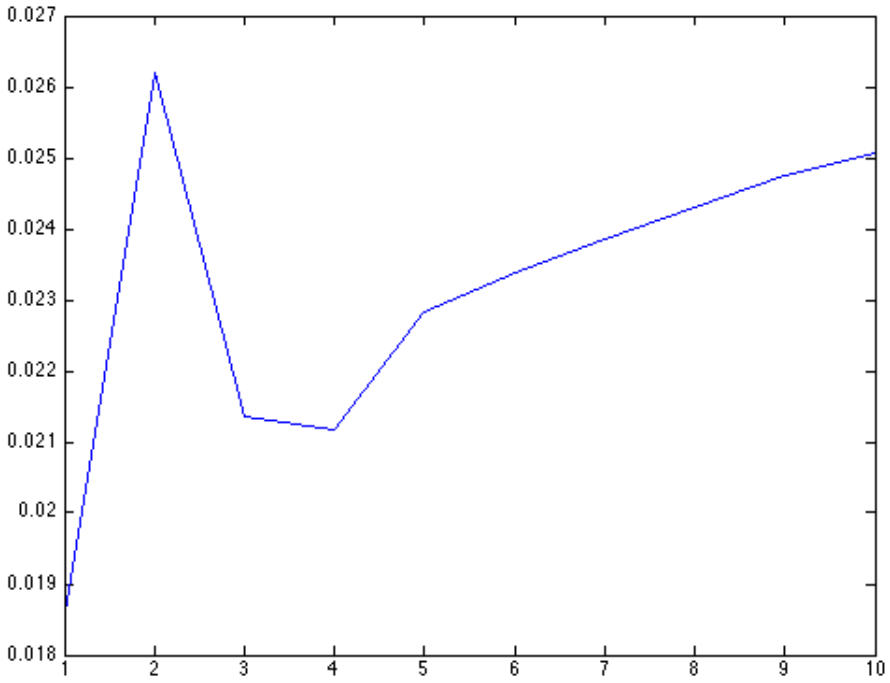
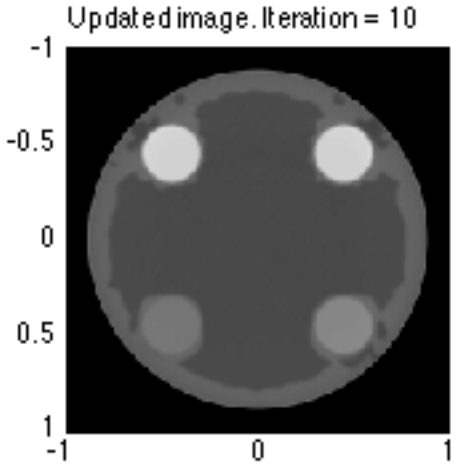


Figure 3.10: ISP reconstructed image after 10 iterations



Chapter 4

Discussion

Figure 4.1 shows the reconstructed images obtained from each of the methods for the modified phantom described in Figure 3.1. Figure 4.2 shows the corresponding density profiles for each of the reconstructed images. All of the methods eliminate the streaking artifact. Additionally, as seen in the density profiles, each method effectively reduces the cupping artifact. However, the Full IGR, the Approximated IGR, and the IFR methods all show substantially higher values at the outer edge of the object. The Full IGR, Approximated IGR, and the ISP methods also all have a thicker ring of incorrectly segmented material around the outer edge. This incorrect segmentation could be due to the algorithm used to minimize the cost function. Thus, this error could possibly be corrected with the use of an improved minimization algorithm. With both IGR methods, this error improved until the cost function reached a minimum. This required approximately 20 iterations with the Full IGR method, as seen in Figure 3.3. The Approximated IGR method required approximately 90 iterations to reach a minimum, as seen in Figure 3.5. As described previously, the approximation leading to the density update in the Approximated IGR method [Eq. (2.22)] decreases the step size taken in each iteration. Therefore, it would be reasonable to expect the Approximated IGR method to require more iterations to reach a minimum than the Full IGR method. The ISP method does not show the same improvement with each iteration. The image improves as the cost function decreases; however, the cost function is not guaranteed to decrease with the ISP method. As the cost function increases, as seen in Figure 3.9, the quality of the reconstructed image declines. In comparison to the other three methods, the ISP method produces the reconstructed image with the poorest resolution. All four methods are able to make a distinction between the two low-contrast materials.

Figure 4.1: Reconstructed images of the modified phantom in Figure 3.1 from each method, **Top left:** Reconstruction from Full IGR method at 20 iterations with grey scale window [0.13,0.35], **Top right:** Reconstruction from Approximated IGR method at 94 iterations with grey scale window [0.13,0.35], **Bottom left:** Reconstruction from IFR method at 4 iterations with grey scale window [0.13,0.35], **Bottom right:** Reconstruction from ISP method at 4 iterations with grey scale window [0.15,0.24].

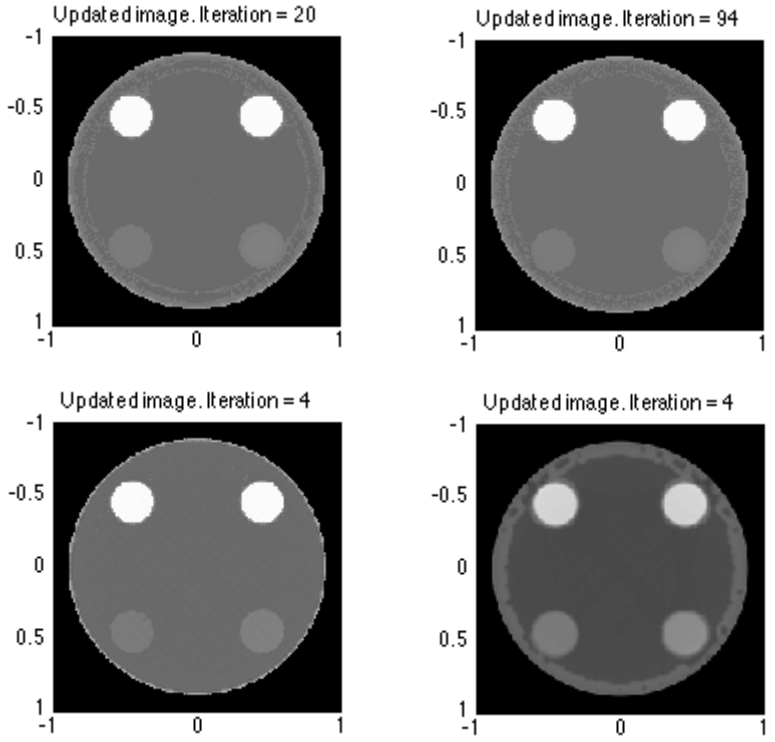
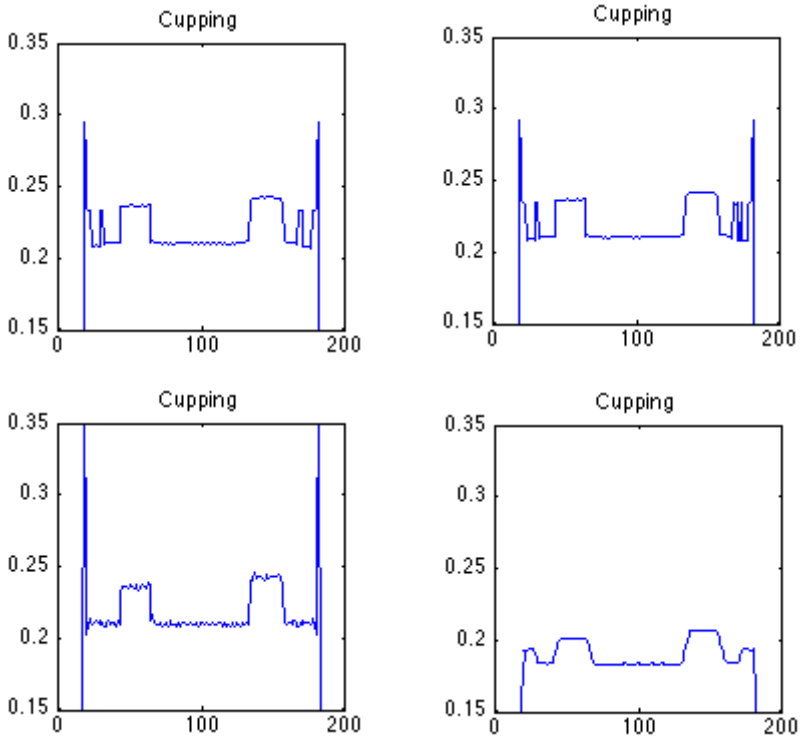


Figure 4.2: Density Profiles for row 135 of the reconstructed images in Figure 4.1, **Top left:** Density profile for Full IGR method , **Top right:** Density profile for Approximated IGR method , **Bottom left:** Density profile for IFR method, **Bottom right:** Density profile for ISP method.



The density update step of the Full IGR method is described as "the main computational bottleneck" by G. Van Gompel, et al [15], motivating the Approximated IGR method. As described in Section 3.2, we found that the method required substantially more time for the segmentation update (60 % of the total time) than it did for the density update. In an effort to reduce the time required for the segmentation update, we limit the number of steps that can be taken in the direction of steepest descent with each iteration. Because the method depends on good initial estimates, we allow the first iteration to take as many steps as needed to minimize the cost function for the segmentation parameter. For each iteration that follows, we limit the number of steps to be taken. Figure 4.3 shows the reconstructed image at 20 iterations with the segmentation limited to 5 steps. Figure 4.4 shows the corresponding density profile for 40 iterations. From the density profile, we see that the cost function decreases by very little after approximately 20 iterations. Thus, the

cost function is still able to be minimized. The segmentation update still requires a majority of the total implementation time, approximately 56%. However, the reconstructed image that we obtain is much closer to the original image than the image obtained before, without the limit placed on the segmentation steps. As described previously, one drawback of this method is that the cost function does not have a unique minimum. This could explain why different images are obtained when we alter the segmentation algorithm but the cost function is still minimized.

Figure 4.3: Full IGR method (with segmentation limit) after 20 iteration, **Top left:** Reconstruction of measured polychromatic data, **Top right:** Density profile for row 135 of polychromatic reconstruction, **Bottom left:** Segmented image, **Bottom right:** Reconstructed image at 20 iterations.

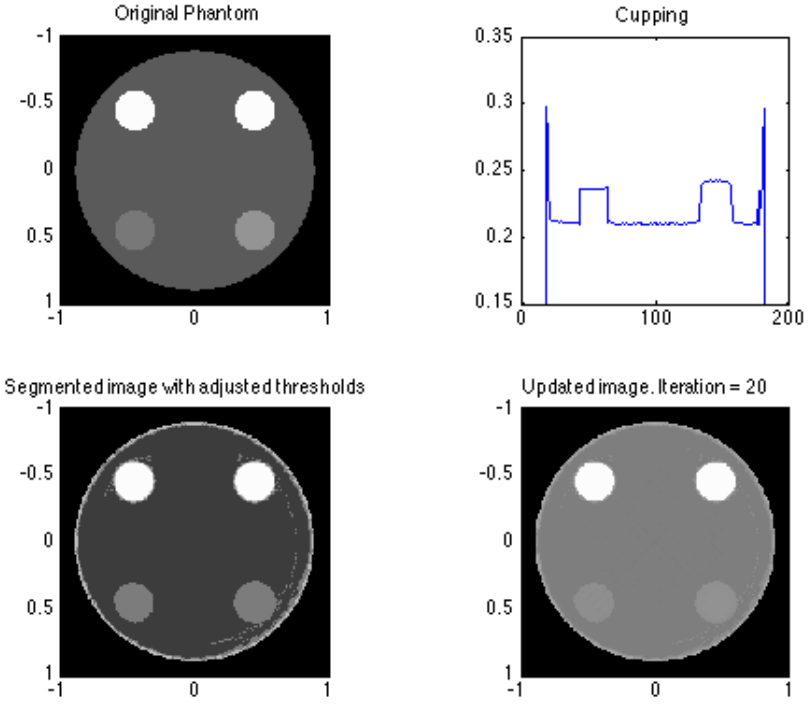
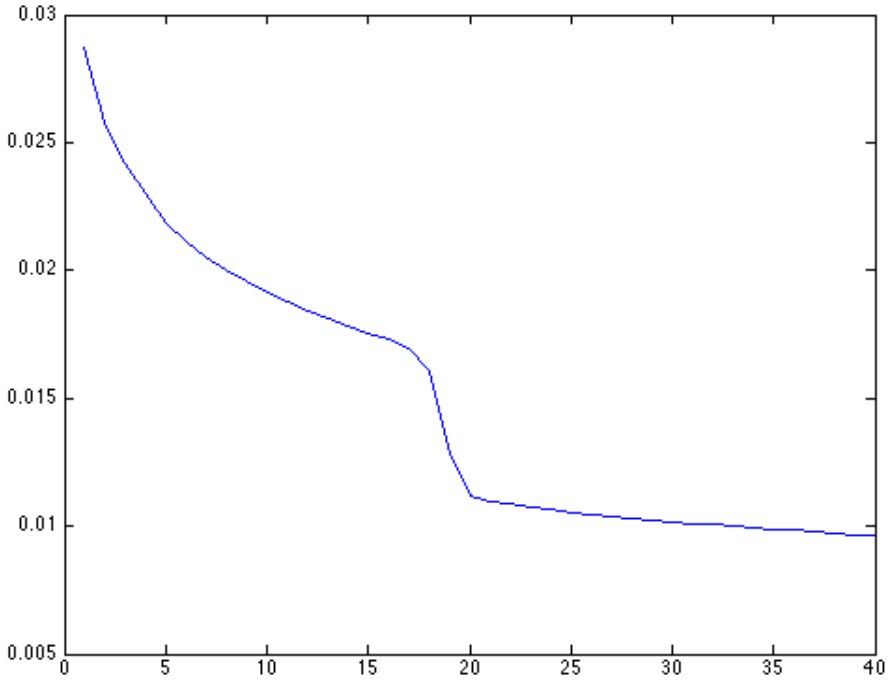


Figure 4.4: Full IGR (with segmentation limit) Cost Function values vs. Iteration



In conclusion, we implemented four iterative reconstruction algorithms to eliminate beam hardening artifacts: the Full IGR method, the Approximated IGR method, the IFR method, and the ISP method. All methods aim to produce a reconstructed image free of artifacts by minimizing the mean square error between the measured polychromatic data and the simulated polychromatic data. Unlike many other methods, those presented here only assume minimal prior knowledge. They assume that the number of materials present in the object is known and that the energy spectrum can be represented by a small, predefined number of energy "bins". Both of the IGR methods as well as the IFR method use a relative density parameter to account for small non-uniformities in the materials. The ISP method assumes that the materials are uniform. All methods were able to correct for the beam hardening artifacts. While the ISP methods was fast, it produced an image with very low resolution and did not improve with further iterations due to the fact that the cost function is not guaranteed to decrease. The IFR method converged to a minimum very quickly (5 iterations) and produced a relatively sharp, accurate image. The IFR method, the

Full IGR method, and the Approximated IGR method were all able to distinguish between the two low-contrast materials using the relative density parameter despite them being segmented incorrectly. In both the Full IGR method as well as the Approximated IGR method, the segmentation update required the greatest amount of time, unlike what is suggested by G. Van Gompel, et al. However, with an improved segmentation algorithm, this could change. Additionally, an improved segmentation algorithm could lead to improved results in both IGR methods, particularly with the material around the edge of the object which is incorrectly segmented in the reconstruction. While the Approximated IGR method is able to update the density faster than the Full IGR method, the decrease in the step size means that more iterations are required to reach a minimum in the cost function. Another drawback to these methods is the non uniqueness of the minimizer of the cost function. This makes it possible for the cost function to be minimized without the reconstructed image replicating the original image. For this reason, it is important to obtain good estimates for each parameter at the beginning of each method.

Appendix A

Full IGR Code

Source code for the Full IGR method.
All code provided by Maryam Alarfaj [1] and edited where noted.

IGRmethodFull.m

```
1  global RFsim
2  global RFpoly  % the measured polychromatic data
3  global Nmat    % number of materials
4  global Npic     % number of pixel
5  global pview    % # of views
6  global Q         % # of rays
7  global Ie
8  global mu
9  global d
10 global Pcurrent
11
12 itmax = 10;
13 Npic = 200;
14 [Q,pview] = size(radon(zeros(Npic,Npic)));
15 Nmat = 5;
16
17 MX=200 ;  MY = 200; %matrix dimensions
18 roi=[-1  1 -1  1]; %roi=[xmin xmax ymin ymax]
19 %region of interest where
20 %reconstruction is computed
21 circle = 1; % If circle = 1 image computed only inside
22 % circle inscribed in roi.
23
24 wmin = 0.17; wmax = 0.25;% the minimum and maximum of window3 to get a
       clear picture
25
```

```
26
27 %Specify parameters of ellipses for mathematical phantom.
28
29 % xe = vector of x-coordinates of centers
30 % ye = vector of y-coordinates of centers
31 % ae = vector of first half axes
32 % be = vector of second half axes
33 % alpha = vector of rotation angles (degrees)
34 %rho = vector of densities [mat1, mat2]
35 xe=[-0.24 0.24 ];
36 ye=[-0.24 0.24];
37 ae=[0.5 0.4]; %minor ax.
38 be=[0.6 0.5]; %major ax.
39 alpha=[0 0];
40
41
42 Ie =[0.1; 0.3; 0.3; 0.2; 0.1]; %1 for monochromatic
43 alpha = alpha*pi/180; % to convert (alpha in degrees) to (radian)
44
45
46 rho =[0.999 -0.7340 ;...
47     0.595 -0.3690 ;...
48     0.416 -0.206 ;...
49     0.265 -0.0820 ;...
50     0.208 -0.0340 ];
51
52 rhosoft = rho(:,1) + rho(:,2); % soft tissue attenuation values
53 rhodiff = -rho(:,2); % rhosoft + rhodiff gives bone attenuation values
54
55 density = [rhosoft rhodiff/8 rhodiff rhodiff rhodiff/4].';
56
57 mu = zeros(length(Ie), Nmat);
58 mu(:,1)=0; %air
59 mu(:,2)=rhosoft; % atten coeff for brain
60 mu(:,3) = rhosoft+rhodiff/8; % atten coeff for soft tissue 1
61 mu(:,4) = rhosoft+rhodiff/4; % atten coeff for soft tissue 2
62 mu(:,5) = rhosoft+rhodiff; % attenuation coeff for bone
63
64
65 Ne = length(Ie);
66 muav = mu(ceil(Ne/2),:)
67 mumax  = max(mu); mumax(mumax==0)=1;
68
69
70
71
```

```
Ellpar =[.9    .15    .15   .15   .15;
          .9    .15    .15   .15   .15;
           0   -.45   -.45   .45   .45;
           0   -.45    .45   .45  -.45;
           0     0      0     0     0].';

%----------The polychromatic data--------

pview = 180; % number of view angels
theta = [0:pview-1]; % direction of the view
xp = [-143:143]/100; % the coordinate of the detector along the line=s
     in radon_ell
RFpoly = zeros(length(xp),length(theta));
for j = 1:length(Ie)
    E = [density(:,j) Ellpar];
    RFpoly = RFpoly + Ie(j)* exp(-(Npic/2)*radon_ell(E,theta,xp));
end
RFpoly = -log(RFpoly); % the uncorrected polychromatic data

I = iradon(RFpoly,[0:179],'linear','Hamming',1,Npic); % iradon is used
    to reconstruct a pic.

%----------------Original Phantom-----------------
 figure;
 E = [density(:,3) Ellpar];
 P = phantom(E,200);
 window3(0.13,0.35,roi,P);title (['Original Phantom']);

Pcurrent = I;
figure;
subplot(221)
window3(wmin,wmax,roi,Pcurrent); title(['Reconstruction from
    polychromatic data']);
window3(wmin,wmax,roi,I); title(['Reconstruction from polychromatic
    data']);
[t1,Ftmp,Dtmp,xdens,Nmat1]= findthreshold2(Pcurrent);

if Nmat1 < Nmat
    t = zeros(Nmat-1,1);
    t(1: Nmat1-2) = t1(1:Nmat1-2);
    t(Nmat1-1:Nmat-2) = t1(Nmat1-2)+ ([1:Nmat-Nmat1]/(1+Nmat-Nmat1))*(
        t1(Nmat1-1)-t1(Nmat1-2));
    t(Nmat-1) = t1(Nmat1-1);
else
    t(1:Nmat-1) = t1(1:Nmat-1);
end
```

```
113       SE = threshold(Pcurrent,t);
114
115
116   subplot(222);
117   window3(1,Nmat,roi,SE); title(['Initial segmented image. Nmat = '
          num2str(Nmat) ' materials.']);
118
119   d= ones(size(SE));
120   C = costfun(mu,Ie,d,SE)
121
122
123   [SE1,phi1,t1] =optims3(t);
124
125   SE = SE1; t = t1;
126
127   subplot(223)
128   window3(1,Nmat,roi,SE); title(['Segmented image with adjusted
          thresholds']);
129
130   % ---------- DENSITY UPDATE WRITTEN BY C.HALL ----------
131   En = length(Ie);
132   [RFsim,Pie,RFspi] = simdatedit(mu,Ie,d,SE);
133   stmp2 = zeros(size(SE));
134   for m = 1:Nmat
135       stmp2(SE==m) = muav(m);
136   end
137   BLint = lineint(mu,Ie,SE,Pie);
138   Bottom = zeros(size(SE));
139   Msumt = zeros(size(SE));
140   for e = 1:En % sum over e in numerator
141       for m = 1:Nmat
142           Msumt(SE==m) = mu(e,m); % sum over m of M(m,e)*s(m,j)
143       end
144       Pie1 = zeros(size(RFpoly));
145       Pie1(:,:) = Pie(e,:,:);
146       Bbpi = BLint.*Pie1./(RFspi.^2);
147       Bbp = iradon(Bbpi,theta,'linear','none',1,Npic); % sum over i in
              numerator
148       Bottom = Bottom + Msumt.*Bbp;
149   end
150   chi = (Bottom == 0);
151   Bottomtmp = ones(size(Bottom));
152   Bottom(chi) = Bottomtmp(chi);
153
154   Top = zeros(size(SE));
155   for e = 1:En % sum over e in numerator
```

```
156       Pie1 = zeros(size(RFpoly));
157       Pie1(:,:) = Pie(e,:,:);
158       Tbpi = (RFpoly-RFsim).*Pie1./RFspi; % (log(I_p,i/I_0) - log(P_i))*
              P_i,e/P_i
159       Tbp = iradon(Tbpi,theta,'linear','none',1,Npic); % sum over i in
              numerator
160       Top = Top + Msumt.*Tbp;
161   end
162
163   Update = Top./Bottom; % density update
164
165   d1 = d + Update;
166   count = 0;
167   C1 = costfun(mu,Ie,d1,SE);
168   while C1 < C
169       count = count+1
170       d = d1;
171       C = C1;
172       d1 = d1 + Update;
173       C1 = costfun(mu,Ie,d,SE);
174   end
175
176   dtmp = (d >= 0);
177   d = d.*dtmp;
178   Pcurrent = d.*stmp2;
179   subplot(224);
180   window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image. 
          Iteration = 1']);
181   % ---------- END DENSITY UPDATE ----------
182
183
184   CF = zeros(itmax,1);
185   CF(1) = costfun(mu,Ie,d,SE)
186
187   t = muav(2:end);
188   for it = 2:itmax
189       it
190       [SE1,phi1,t1] =optims3limit(t,3); % to limit steps <==========
              EDITED BY C.HALL
191       %[SE1,phi1,t1] =optims3(t);
192
193       SE = SE1; t = t1;
194       figure
195       subplot(223)
196       window3(1,Nmat,roi,SE); title(['Segmented image with adjusted 
              thresholds']);
```

```
197
198     % ———————— DENSITY UPDATE WRITTEN BY C.HALL ————————
199     En = length(Ie);
200     [RFsim,Pie,RFspi] = simdatedit(mu,Ie,d,SE);
201     stmp2 = zeros(size(SE));
202     for m = 1:Nmat
203         stmp2(SE==m) = muav(m);
204     end
205     BLint = lineint(mu,Ie,SE,Pie);
206     Bottom = zeros(size(SE));
207     Msumt = zeros(size(SE));
208     for e = 1:En % sum over e in numerator
209         for m = 1:Nmat
210             Msumt(SE==m) = mu(e,m); % sum over m of M(m,e)*s(m,j)
211         end
212         Pie1 = zeros(size(RFpoly));
213         Pie1(:,:) = Pie(e,:,:);
214         Bbpi = BLint.*Pie1./(RFspi.^2);
215         Bbp = iradon(Bbpi,theta,'linear','none',1,Npic); % sum over i
                in numerator
216         Bottom = Bottom + Msumt.*Bbp;
217     end
218     chi = (Bottom == 0);
219     Bottomtmp = ones(size(Bottom));
220     Bottom(chi) = Bottomtmp(chi);
221
222
223     Top = zeros(size(SE));
224     for e = 1:En % sum over e in numerator
225
226         Pie1 = zeros(size(RFpoly));
227         Pie1(:,:) = Pie(e,:,:);
228         Tbpi = (RFpoly-RFsim).*Pie1./RFspi; % (log(I_p,i/I_0) - log(P_i
                ))*P_i,e/P_i
229         Tbp = iradon(Tbpi,theta,'linear','none',1,Npic); % sum over i
                in numerator
230         Top = Top + Msumt.*Tbp;
231     end
232
233     Update = Top./Bottom; % density update
234
235     d1 = d + Update;
236     count = 0;
237     C1 = costfun(mu,Ie,d1,SE);
238     while C1 < C
239         count = count+1
```

```
240             d = d1;
241             C = C1;
242             d1 = d1 + Update;
243             C1 = costfun(mu,Ie,d,SE);
244         end
245
246         dtmp = (d >= 0);
247         d = d.*dtmp;
248         Pcurrent = d.*stmp2;
249         subplot(224);
250         window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image. Iteration = ' num2str(it)]);
251         % ———— END DENSITY UPDATE ————
252
253         subplot(221)
254         window3(0.13,0.35,roi,P);title ('Original Phantom');
255         subplot(222)
256         plot(Pcurrent(135,:)); title('Cupping'); axis('square');axis([0 200 0.15 0.35]);
257
258         CF(it) = costfun(mu,Ie,d,SE)
259 end
260
261 its = [1:itmax];
262 figure;
263 plot(its,CF)
```

radon_ell.m

```
1
2  function [RF] = radon_ell(E,theta,s)
3
4 %  This function computes the Radon transform of ellipses
5 % Input: E  as for MATLAB function phantom
6 %          theta as for MATLAB function radon
7 %         s: vector with radial coordinates corresponding to each row of RF
8 %
9 rho = E(:,1); u = E(:,2); v = E(:,3);
10 x = E(:,4); y = E(:,5); alpha = E(:,6);
11
12 ne = length(rho);
13 RF = zeros(length(s),length(theta));
14
15 for j = 1:length(theta);
16     phi = pi*theta(j)/180;
17     omega = [cos(phi);sin(phi)];
```

```
18      tmp =zeros(1,length(s));
19      for mu = 1:ne
20          a = (u(mu)*cos(phi-alpha(mu)))^2+(v(mu)*sin(phi-alpha(mu)))^2;
21          test = a-(s-[x(mu);y(mu)]'*omega).^2;
22          ind = test>0;
23          tmp(ind) = tmp(ind)+rho(mu)*(2*u(mu)*v(mu)*sqrt(test(ind)))/a;
24      end  % mu-loop
25      RF(:,j) = tmp.';
26  end
```

window3.m

```
1   function pic1 = window3(mi,ma,roi,pic);
2   %function pic1 = window3(mi,ma,roi,pic);
3   % displays image pic with coordinates given by roi
4   % roi = [xmin xmax ymin ymax]
5   x = [roi(1), roi(2)]; y = [roi(3), roi(4)];
6   colors = 128; co = colors-1;
7   pic1 = pic - mi*ones(size(pic));
8   pic1 = (co/(ma-mi))*pic1;
9   P = (pic1 >= 0);
10  pic1 = pic1.*P;
11  P = (pic1 <= co);
12  pic1 = pic1.*P + co*(ones(size(pic1)) - P);
13  colormap(gray(colors));
14  image(x,fliplr(y),flipud(pic1));
15  axis('square');
```

findthreshold2.m

```
1   function [T,F,D,xdens,N] = findthreshold2(P);
2   %Threshold a picture known of containing N materials.
3   % P = picture to the thrsholded
4   % N =  number of materials
5   % T = vector of length N-1 of threshold values.
6   whi = 0.97
7   relthresh = 0.02;
8   Nbin = 50;
9   pmin = min(min(P));
10  pmax = max(max(P));
11  dx = 1.2*(pmax-pmin)/Nbin;
12  xdens = pmin + [0:Nbin]*dx;
13  F = zeros(Nbin+1,1);
14  for n = 0:Nbin
15      F(n+1) = sum(sum(P<= pmin+n*dx));
16  end
17  D= F(2:end)-F(1:end-1);
```

```
18 mph = relthresh*max(D);
19 [pks,locs] = findpeaks(D,'MINPEAKHEIGHT',mph)
20 tmp = [0 (xdens(locs)-dx/2)];
21  T = whi*tmp(2:end) + (1-whi)*tmp(1:end-1);
22  N = length(T)+1;
```

threshold.m

```
1  function S = threshold(P,t)
2  % threshold function
3  % t is threshold value
4
5  S= ones(size(P));
6
7  Nmat = length(t)+1;
8  for j=1:Nmat-2
9       chi = ((P >= t(j))&(P < t(j+1)));
10      S(chi) = j+1;
11 end
12 chi = P >= t(end); % end is the last material
13 S(chi) = Nmat;
```

costfun.m

```
1  function phi = costfun(mu,Ie,d,s)
2  % phi is the cost function that we want to minimize
3  % mu is the attenuation coef.
4  % I is the energy spectrum
5  % d is the relative density which models the small variation in
       attenuation
6  % within one material
7  global Pcurrent
8  global RFsim
9  global RFpoly  % the measured polychromatic data
10 global Nmat   % number of materials
11 global Npic    % number of pixel
12 global pview   % # of views
13 global Q       % # of rays
14
15
16 RFsim = simdat(mu,Ie,d,s);
17
18 phi = (norm(RFpoly - RFsim,'fro')^2)/(pview*Q); % phi is the square of
       the Frobenius norm of the difference btween the measured and the
       simulated devided by the # of data
19
20 end
```

optims3.m

```
function [SE,phi,t] =optims3(t0)

global mu
global Ie
global d
global Pcurrent


dtfac = 0.1;
Nt = length(t0);
dt = zeros(Nt,1);
SE0 = threshold(Pcurrent,t0);
phi0 = costfun(mu,Ie,d,SE0);

t = zeros(Nt,1); t(1:Nt) = t0(1:Nt);
t
dt(1) = dtfac*t(1);
for k = 2:Nt
    dt(k) = dtfac*(t(k)-t(k-1));
end
gradcost = zeros(Nt,1);
Id = eye(Nt);
for k = 1:Nt
    gradcost(k) = (costfunt(mu,Ie,d,t+.5*dt(k)*Id(:,k))- ...
        costfunt(mu,Ie,d,t-.5*dt(k)*Id(:,k)))/dt(k);
end
ng = norm(gradcost);

SE=SE0; phi = phi0;
if ng >= 1.e-6
    gradunit = gradcost/ng;
    dtv = min(dt);
    t;
    dtv*gradunit;
    t1 = t - dtv*gradunit;
    phi1 = costfunt(mu,Ie,d,t1)
    while phi1< phi0
        t = t1;
        phi0 = phi1;
        t1 = t1 - dtv*gradunit;
        phi1 = costfunt(mu,Ie,d,t1);
    end
    SE = threshold(Pcurrent,t);
    phi = phi0;
end
```

simdatedit.m

```
1  function [RFs,Pie,RFspi] = simdatedit(mu,Ie,d,s)
2  % phi is the cost function that we want to minimize
3  % mu is the attenuation coef.
4  % I is the energy spectrum
5  % d is the relative density which models the small variation in
       attenuation
6  % within one material
7  global Pcurrent
8
9  global RFpoly  % the measured polychromatic data
10 global Nmat   % number of materials
11 global Npic    % number of pixel
12 global pview   % # of views
13 global Q       % # of rays
14
15
16 %P =zeros(Npic,Npic); % instead of have a stack of images we use a
       single image with three different values (s==n) to represent the
       three materials
17
18 RP=zeros(Nmat,Q,pview);
19 Pie = RP; %<========== EDITED BY C.HALL
20 for n = 1: Nmat  % this loop for every n materials
21     P =zeros(Npic,Npic); % instead of have a stack of images we use a
           single image with three different values (s==n) to represent the
           three materials
22     chi = (s==n); % find the pixels in the image P where the s =
           material n
23     P(chi) = d(chi); % this is the dj*snj
24     RP(n,:,:)=radon(P); %the radon data of the picture P(chi) which is
           dj*snj
25 end
26
27
28 RFs = zeros(size(RFpoly));% initialize a matrix with the simulated data
29 RFspi = RFs; %<========== EDITED BY C.HALL
30
31 En = length(Ie);
32 for e = 1:En % outter loop for energy
33     tmp = zeros(size(RFpoly)); % temporary variable which represent the
           n sum
34     for n=1:Nmat % inner loop for the material
35         R1= zeros(size(RFpoly));
36         R1(:,:) = RP(n,:,:); % R1 is used instead of RP, to write the
               RP in term of 2 indesis instead of 3
```

```
37          tmp = tmp + mu(e,n)*R1; % the sum over n
38          Pie(e,:,:) = Ie(e)*exp(-tmp); %<========== EDITED BY C.HALL
39      end
40      RFspi = RFspi + Ie(e)*exp(-tmp); % to compute the e sum
41  end
42  RFs = -log(RFspi); % the simulated polychromatic data
```

lineint.m

```
1  % ------ CODE ADAPTED FROM simdat.m BY C.HALL ------
2  function Lint = lineint(mu,Ie,s,Pie)
3  % phi is the cost function that we want to minimize
4  % mu is the attenuation coef.
5  % I is the energy spectrum
6  % d is the relative density which models the small variation in
       attenuation
7  % within one material
8  global RFpoly  % the measured polychromatic data
9  global Nmat    % number of materials
10 global Npic     % number of pixel
11 global pview    % # of views
12 global Q        % # of rays
13
14
15
16 RP=zeros(Nmat,Q,pview);
17 for n = 1: Nmat  % this loop for every n materials
18      P =zeros(Npic,Npic); % instead of have a stack of images we use a
            single image with three different values (s==n) to represent the
             three materials
19      Itmp = ones(Npic,Npic);
20      chi = (s==n); % find the pixels in the image P where the s =
            material n
21      P(chi) = Itmp(chi);
22      RP(n,:,:) = radon(P);
23 end
24
25 Lint = zeros(size(RFpoly));% initialize a matrix with the line integral
26
27 En = length(Ie);
28 for e = 1:En % outter loop for energy
29      tmp = zeros(size(RFpoly)); % temporary variable which represent the
             n sum
30      for n=1:Nmat % inner loop for the material
31          R1= zeros(size(RFpoly));
32          R1(:,:) = RP(n,:,:); % R1 is used instead of RP, to write the
                RP in term of 2 indices instead of 3
```

```
33          tmp = tmp + mu(e,n)*R1; % the sum over n
34      end
35      Pie1 = zeros(size(RFpoly));
36      Pie1(:,:) = Pie(e,:,:);
37      Lint = Lint + Pie1.*tmp; % to compute the e sum
38  end
```

optims3limit.m

```
1   function [SE,phi,t] =optims3limit(t0,n)
2
3   global mu
4   global Ie
5   global d
6   global Pcurrent
7
8
9   dtfac = 0.1;
10  Nt = length(t0);
11  dt = zeros(Nt,1);
12  SE0 = threshold(Pcurrent,t0);
13  phi0 = costfun(mu,Ie,d,SE0);
14
15  t = zeros(Nt,1); t(1:Nt) = t0(1:Nt);
16  t
17  dt(1) = dtfac*t(1);
18  for k = 2:Nt
19      dt(k) = dtfac*(t(k)-t(k-1));
20  end
21  gradcost = zeros(Nt,1);
22  Id = eye(Nt);
23  for k = 1:Nt
24      gradcost(k) = (costfunt(mu,Ie,d,t+.5*dt(k)*Id(:,k))- ...
25          costfunt(mu,Ie,d,t-.5*dt(k)*Id(:,k)))/dt(k);
26  end
27  ng = norm(gradcost);
28
29  SE=SE0; phi = phi0;
30  if ng >= 1.e-6
31      gradunit = gradcost/ng;
32      dtv = min(dt);
33      t;
34      dtv*gradunit;
35      t1 = t - dtv*gradunit;
36      phi1 = costfunt(mu,Ie,d,t1)
37      it = 0; % <========== EDITED BY C.HALL
38      while phi1< phi0 & it<n % <========== EDITED BY C.HALL
```

```
39          it = it+1; % <========== EDITED BY C.HALL
40          t = t1;
41          phi0 = phi1;
42          t1 = t1 - dtv*gradunit;
43          phi1 = costfunt(mu,Ie,d,t1);
44      end
45      SE = threshold(Pcurrent,t);
46      phi = phi0;
47  end
```

costfunt.m

```
1   function phi = costfunt(mu,Ie,d,t)
2   % phi is the cost function that we want to minimize
3   % mu is the attenuation coef.
4   % I is the energy spectrum
5   % d is the relative density which models the small variation in
        attenuation
6   % within one material
7   global Pcurrent
8   global RFsim
9   global RFpoly  % the measured polychromatic data
10  global Nmat    % number of materials
11  global Npic    % number of pixel
12  global pview   % # of views
13  global Q       % # of rays
14
15  s = threshold(Pcurrent,t);
16
17
18  RP=zeros(Nmat,Q,pview);
19  for n = 1: Nmat  % this loop for every n materials
20      P =zeros(Npic,Npic); % instead of have a stack of images we use a
            single image with three different values (s==n) to represent the
             three materials
21      chi = (s==n); % find the pixels in the image P where the s =
            material n
22      P(chi) = d(chi); % this is the dj*snj
23      RP(n,:,:)=radon(P); %the radon data of the picture P(chi) which is
            dj*snj
24  end
25
26
27  RFsim = zeros(size(RFpoly));% initialize a matrix with the simulated
        data
28
29
```

```
30 En = length(Ie);
31 for e = 1:En % outter loop for energy
32     tmp = zeros(size(RFpoly)); % temporary variable which represent the
           n sum
33     for n=1:Nmat % inner loop for the material
34         R1= zeros(size(RFpoly));
35         R1(:,:) = RP(n,:,:); % R1 is used instead of RP, to write the
               RP in term of 2 indesis instead of 3
36         tmp = tmp + mu(e,n)*R1; % the sum over n
37     end
38     RFsim = RFsim + Ie(e)*exp(-tmp); % to compute the e sum
39 end
40 RFsim = -log(RFsim); % the simulated polychromatic data
41 phi = (norm(RFpoly - RFsim,'fro')^2)/(pview*Q); % phi is the square of
       the Frobenius norm of the difference btween the measured and the
       simulated devided by the # of data
42 maxRFpoly = max(max(RFpoly));
43 maxRFsim = max(max(RFsim));
44 end
```

Appendix B

Approximated IGR Code

Source code for the Approximated IGR method.
All code provided by Maryam Alarfaj [1] and edited where noted.

IGRmethodApprox.m

```
1  global RFsim
2  global RFpoly  % the measured polychromatic data
3  global Nmat    % number of materials
4  global Npic     % number of pixel
5  global pview    % # of views
6  global Q         % # of rays
7  global Ie
8  global mu
9  global d
10 global Pcurrent
11
12 itmax = 10;
13 Npic = 200;
14 [Q,pview] = size(radon(zeros(Npic,Npic)));
15 Nmat = 5;
16
17 MX=200 ;  MY = 200; %matrix dimensions
18 roi=[-1  1 -1  1]; %roi=[xmin xmax ymin ymax]
19 %region of interest where
20 %reconstruction is computed
21 circle = 1; % If circle = 1 image computed only inside
22 % circle inscribed in roi.
23
24 wmin = 0.17; wmax = 0.25;% the minimum and maximum of window3 to get a
       clear picture
25
```

```
26
27 %Specify parameters of ellipses for mathematical phantom.
28
29 % xe = vector of x-coordinates of centers
30 % ye = vector of y-coordinates of centers
31 % ae = vector of first half axes
32 % be = vector of second half axes
33 % alpha = vector of rotation angles (degrees)
34 %rho = vector of densities [mat1, mat2]
35 xe=[-0.24 0.24 ];
36 ye=[-0.24 0.24];
37 ae=[0.5 0.4]; %minor ax.
38 be=[0.6 0.5]; %major ax.
39 alpha=[0 0];
40
41 Ie =[0.1; 0.3; 0.3; 0.2; 0.1]; %1 for monochromatic
42 alpha = alpha*pi/180; % to convert (alpha in degrees) to (radian)
43
44 rho =[0.999 -0.7340 ;...
45     0.595 -0.3690 ;...
46     0.416 -0.206 ;...
47     0.265 -0.0820 ;...
48     0.208 -0.0340 ];
49
50  rhosoft = rho(:,1) + rho(:,2); % soft tissue attenuation values
51  rhodiff = -rho(:,2); % rhosoft + rhodiff gives bone attenuation values
52
53 density = [rhosoft rhodiff/8 rhodiff rhodiff rhodiff/4].';
54
55 mu = zeros(length(Ie), Nmat);
56 mu(:,1)=0; %air
57 mu(:,2)=rhosoft; % atten coeff for brain
58 mu(:,3) = rhosoft+rhodiff/8; % atten coeff for soft tissue 1
59 mu(:,4) = rhosoft+rhodiff/4; % atten coeff for soft tissue 2
60 mu(:,5) = rhosoft+rhodiff; % attenuation coeff for bone
61
62 mu
63
64 Ne = length(Ie);
65 muav = mu(ceil(Ne/2),:)
66 mumax  = max(mu); mumax(mumax==0)=1;
67
68
69
70
71 Ellpar =[.9   .15   .15   .15   .15;
```

```
72            .9    .15    .15   .15   .15;
73             0   -.45   -.45   .45   .45;
74             0   -.45    .45   .45  -.45;
75             0     0      0     0     0].';
76
77 %----------The polychromatic data-------
78
79 pview = 180; % number of view angels
80 theta = [0:pview-1]; % direction of the view
81 xp = [-143:143]/100; % the coordinate of the detector along the line=s
       in radon_ell
82 RFpoly = zeros(length(xp),length(theta));
83 for j = 1:length(Ie)
84     E = [density(:,j) Ellpar];
85     RFpoly = RFpoly + Ie(j)* exp(-(Npic/2)*radon_ell(E,theta,xp));
86 end
87 RFpoly = -log(RFpoly); % the uncorrected polychromatic data
88
89 I = iradon(RFpoly,[0:179],'linear','Hamming',1,Npic); % iradon is used
      to reconstruct a pic.
90
91 %----------------Original Phantom----------------
92  figure;
93  E = [density(:,3) Ellpar];
94  P = phantom(E,200);
95  window3(0.13,0.35,roi,P);title (['Original Phantom']);
96
97
98 Pcurrent = I;
99 figure;
100 subplot(221)
101 window3(wmin,wmax,roi,Pcurrent); title(['Reconstruction from
      polychromatic data']);
102 window3(wmin,wmax,roi,I); title(['Reconstruction from polychromatic
      data']);
103 [t1,Ftmp,Dtmp,xdens,Nmat1]= findthreshold2(Pcurrent);
104
105 if Nmat1 < Nmat
106     t = zeros(Nmat-1,1);
107     t(1: Nmat1-2) = t1(1:Nmat1-2);
108     t(Nmat1-1:Nmat-2) = t1(Nmat1-2)+ ([1:Nmat-Nmat1]/(1+Nmat-Nmat1))*(
          t1(Nmat1-1)-t1(Nmat1-2));
109     t(Nmat-1) = t1(Nmat1-1);
110 else
111     t(1:Nmat-1) = t1(1:Nmat-1);
112 end
```

```
113       SE = threshold(Pcurrent,t);
114
115
116   subplot(222);
117   window3(1,Nmat,roi,SE); title(['Initial segmented image. Nmat = '
          num2str(Nmat) ' materials.']);
118
119   d= ones(size(SE));
120   C = costfun(mu,Ie,d,SE)
121
122
123   [SE1,phi1,t1] =optims3(t);
124
125   SE = SE1; t = t1;
126
127   subplot(223)
128   window3(1,Nmat,roi,SE); title(['Segmented image with adjusted
          thresholds']);
129
130
131   % ---------- DENSITY UPDATE WRITTEN BY C.HALL ----------
132   En = length(Ie);
133   [RFsim,Pie,RFspi] = simdatedit(mu,Ie,d,SE);
134   Mesum = sum(mu); % sum over energies(e) of mu(m,e)
135   Msumb = zeros(size(SE));
136   stmp2 = zeros(size(SE));
137   for m = 1:Nmat
138       Msumb(SE==m) = Mesum(m); % sum over m of sum over e of mu(m,e)*s(m,
              j)
139       stmp2(SE==m) = muav(m);
140   end
141   Lint = lineint(mu,Ie,SE,Pie);
142   Ird = iradon(Lint./RFspi,theta,'linear','none',1,Npic);
143   denom = Msumb.*Ird; % computing denominator of density approx.
144   chi = (denom == 0);
145   denomtmp = ones(size(denom));
146   denom(chi) = denomtmp(chi);
147
148   Msumt = zeros(size(SE));
149   Top = zeros(size(SE));
150   for e = 1:En % sum over e in numerator
151       for m = 1:Nmat
152           Msumt(SE==m) = mu(e,m); % sum over m of M(m,e)*s(m,j)
153       end
154       Pie1 = zeros(size(RFpoly));
155       Pie1(:,:) = Pie(e,:,:);
```

```
156       Tbpi = (RFpoly-RFsim).*Pie1./RFspi; % (log(I_p,i/I_0) - log(P_i))*
              P_i,e/P_i
157       Tbp = iradon(Tbpi,theta,'linear','none',1,Npic); % sum over i in
              numerator
158       Top = Top + Msumt.*Tbp;
159   end
160
161   Update = Top./denom; % density update
162
163   d1 = d + Update;
164   count = 0;
165   C1 = costfun(mu,Ie,d1,SE);
166   while C1 < C
167       count = count+1
168       d = d1;
169       C = C1;
170       d1 = d1 + Update;
171       C1 = costfun(mu,Ie,d,SE);
172   end
173
174   dtmp = (d >= 0);
175   d = d.*dtmp;
176   Pcurrent = d.*stmp2;
177   time(1) = toc
178   subplot(224);
179   window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image. 
          Iteration = 1']);
180   % ---------- END DENSITY UPDATE ----------
181
182
183   CF = zeros(itmax,1);
184   CF(1) = costfun(mu,Ie,d,SE)
185
186
187   t = muav(2:end);
188   for it = 2:itmax
189       it
190       %[SE1,phi1,t1] =optims3(t);
191       [SE1,phi1,t1] =optims3limit(t,5); % to limit steps <==========
              EDITED BY C.HALL
192
193
194       SE = SE1; t = t1;
195       figure
196       subplot(223)
```

```
197     window3(1,Nmat,roi,SE); title(['Segmented image with adjusted
            thresholds']);
198
199     % ---------- DENSITY UPDATE WRITTEN BY C.HALL ----------
200     En = length(Ie);
201     [RFsim,Pie,RFspi] = simdatedit(mu,Ie,d,SE);
202     Mesum = sum(mu); % sum over energies(e) of mu(m,e)
203     Msumb = zeros(size(SE));
204     stmp2 = zeros(size(SE));
205     for m = 1:Nmat
206         Msumb(SE==m) = Mesum(m); % sum over m of sum over e of mu(m,e)*
                s(m,j)
207         stmp2(SE==m) = muav(m);
208     end
209     Lint = lineint(mu,Ie,SE,Pie);
210     Ird = iradon(Lint./RFspi,theta,'linear','none',1,Npic);
211     denom = Msumb.*Ird; % computing denominator of density approx.
212     chi = (denom == 0);
213     denomtmp = ones(size(denom));
214     denom(chi) = denomtmp(chi);
215
216     Msumt = zeros(size(SE));
217     Top = zeros(size(SE));
218     for e = 1:En % sum over e in numerator
219         for m = 1:Nmat
220             Msumt(SE==m) = mu(e,m); % sum over m of M(m,e)*s(m,j)
221         end
222         Pie1 = zeros(size(RFpoly));
223         Pie1(:,:) = Pie(e,:,:);
224         Tbpi = (RFpoly-RFsim).*Pie1./RFspi; % (log(I_p,i/I_0) - log(P_i
                ))*P_i,e/P_i
225         Tbp = iradon(Tbpi,theta,'linear','none',1,Npic); % sum over i
                in numerator
226         Top = Top + Msumt.*Tbp;
227     end
228
229     Update = Top./denom; % density update
230
231     d1 = d + Update;
232     count = 0;
233     C1 = costfun(mu,Ie,d1,SE);
234     while C1 < C
235         count = count+1
236         d = d1;
237         C = C1;
238         d1 = d1 + Update;
```

```
239             C1 = costfun(mu,Ie,d,SE);
240         end
241
242         dtmp = (d >= 0);
243         d = d.*dtmp;
244         Pcurrent = d.*stmp2;
245         time(it) = toc
246         subplot(224);
247         window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image. Iteration = ' num2str(it)]);
248         % ---------- END DENSITY UPDATE ----------
249
250         subplot(221)
251         window3(0.13,0.35,roi,P);title ('Original Phantom');
252         subplot(222)
253         plot(Pcurrent(135,:)); title('Cupping'); axis('square');axis([0 200 0.15 0.35]);
254
255         CF(it) = costfun(mu,Ie,d,SE)
256     end
257
258     its = [1:itmax];
259     figure;
260     plot(its,CF)
```

Appendix C

IFR Code

Source code for the IFR method.
All code provided by Maryam Alarfaj [1].

IFRmethod.m

```
1  global RFsim
2  global RFpoly  % the measured polychromatic data
3  global Nmat    % number of materials
4  global Npic     % number of pixel
5  global pview    % # of views
6  global Q         % # of rays
7  global Ie
8  global mu
9  global d
10 global Pcurrent
11
12 itmax = 10;
13 Npic = 200;
14 [Q,pview] = size(radon(zeros(Npic,Npic)));
15 Nmat = 5;
16
17 MX=200 ;  MY = 200; %matrix dimensions
18 roi=[-1  1 -1  1]; %roi=[xmin xmax ymin ymax]
19 %region of interest where
20 %reconstruction is computed
21 circle = 1; % If circle = 1 image computed only inside
22 % circle inscribed in roi.
23
24 wmin = 0.17; wmax = 0.25;% the minimum and maximum of window3 to get a
       clear picture
25
```

```
26
27 %Specify parameters of ellipses for mathematical phantom.
28
29 % xe = vector of x-coordinates of centers
30 % ye = vector of y-coordinates of centers
31 % ae = vector of first half axes
32 % be = vector of second half axes
33 % alpha = vector of rotation angles (degrees)
34 %rho = vector of densities [mat1, mat2]
35 xe=[-0.24 0.24 ];
36 ye=[-0.24 0.24];
37 ae=[0.5 0.4]; %minor ax.
38 be=[0.6 0.5]; %major ax.
39 alpha=[0 0];
40
41
42 Ie =[0.1; 0.3; 0.3; 0.2; 0.1]; %1 for monochromatic
43 alpha = alpha*pi/180; % to convert (alpha in degrees) to (radian)
44
45
46 rho =[0.999 -0.7340 ;...
47     0.595 -0.3690 ;...
48     0.416 -0.206 ;...
49     0.265 -0.0820 ;...
50     0.208 -0.0340 ];
51
52 rhosoft = rho(:,1) + rho(:,2); % soft tissue attenuation values
53 rhodiff = -rho(:,2); % rhosoft + rhodiff gives bone attenuation values
54
55 density = [rhosoft rhodiff/8 rhodiff rhodiff rhodiff/4].';
56
57 mu = zeros(length(Ie), Nmat);
58 mu(:,1)=0; %air
59 mu(:,2)=rhosoft; % atten coeff for brain
60 mu(:,3) = rhosoft+rhodiff/8; % atten coeff for soft tissue 1
61 mu(:,4) = rhosoft+rhodiff/4; % atten coeff for soft tissue 2
62 mu(:,5) = rhosoft+rhodiff; % attenuation coeff for bone
63
64 mu
65
66 Ne = length(Ie);
67 muav = mu(ceil(Ne/2),:)
68 mumax  = max(mu); mumax(mumax==0)=1;
69
70
71
```

```
72
73  Ellpar =[.9    .15    .15   .15   .15;
74           .9    .15    .15   .15   .15;
75            0   -.45   -.45   .45   .45;
76            0   -.45    .45   .45  -.45;
77            0     0      0     0     0].';
78  %----------The polychromatic data--------
79
80  pview = 180; % number of view angels
81  theta = [0:pview-1]; % direction of the view
82  xp = [-143:143]/100; % the coordinate of the detector along the line=s
        in radon_ell
83  RFpoly = zeros(length(xp),length(theta));
84  for j = 1:length(Ie)
85      E = [density(:,j) Ellpar];
86      RFpoly = RFpoly + Ie(j)* exp(-(Npic/2)*radon_ell(E,theta,xp));
87  end
88  RFpoly = -log(RFpoly); % the uncorrected polychromatic data
89
90  I = iradon(RFpoly,[0:179],'linear','Hamming',1,Npic); % iradon is used
       to reconstruct a pic.
91
92  figure;
93  E = [density(:,3) Ellpar];
94  P = phantom(E,200);
95  window3(0.13,0.35,roi,P);title (['Original Phantom']);
96
97
98  %----Computing the segmented Image
99  Pcurrent = I;
100 figure;
101 subplot(221)
102 window3(wmin,wmax,roi,Pcurrent); title(['Reconstruction from
        polychromatic data']);
103 window3(wmin,wmax,roi,I); title(['Reconstruction from polychromatic
        data']);
104
105 [t1,Ftmp,Dtmp,xdens,Nmat1]= findthreshold2(Pcurrent);
106
107 if Nmat1 < Nmat
108     t = zeros(Nmat-1,1);
109     t(1: Nmat1-2) = t1(1:Nmat1-2);
110     t(Nmat1-1:Nmat-2) = t1(Nmat1-2)+ ([1:Nmat-Nmat1]/(1+Nmat-Nmat1))*(
          t1(Nmat1-1)-t1(Nmat1-2));
111     t(Nmat-1) = t1(Nmat1-1);
112 else
```

```
113       t(1:Nmat-1) = t1(1:Nmat-1);
114   end
115       SE = threshold(Pcurrent,t);
116
117
118   %---initial guess for a segmented image
119   subplot(222);
120   window3(1,Nmat,roi,SE); title(['Initial segmented image. Nmat = '
          num2str(Nmat) ' materials.']);
121
122   d= ones(size(SE));
123   C = costfun(mu,Ie,d,SE)
124
125
126   [SE1,phi1,t1] =optims3(t);
127
128   SE = SE1; t = t1;
129
130   subplot(223)
131   window3(1,Nmat,roi,SE); title(['Segmented image with adjusted
          thresholds']);
132   RFsim = simdat(mu,Ie,d,SE);
133   Ptmp = iradon(RFpoly-RFsim,[0:179],'linear','Hamming',1,Npic);
134   stmp1 = zeros(size(SE)); stmp2 = stmp1;
135   for k = 1:Nmat
136       stmp1(SE==k) = 1/mumax(k);
137       stmp2(SE==k) = muav(k);
138   end
139   d = d + Ptmp.*stmp1;
140   Pcurrent = d.*stmp2;
141   subplot(224);
142   window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image.
          Iteration = 1' ]);
143   CF = zeros(itmax,1);
144   CF(1) = costfun(mu,Ie,d,SE)
145
146
147   t = muav(2:end);
148   for it = 2:itmax
149       it
150       [SE1,phi1,t1] =optims3(t);
151
152
153       SE = SE1; t = t1;
154       figure
155       subplot(223)
```

```
156       window3(1,Nmat,roi,SE); title(['Segmented image with adjusted
              thresholds']);
157       RFsim = simdat(mu,Ie,d,SE);
158       Ptmp = iradon(RFpoly-RFsim,[0:179],'linear','Hamming',1,Npic);
159       stmp1 = zeros(size(SE)); stmp2 = stmp1;
160       for k = 1:Nmat
161           stmp1(SE==k) = 1/mumax(k);
162           stmp2(SE==k) = muav(k);
163       end
164       d = d + Ptmp.*stmp1;
165       Pcurrent = d.*stmp2;
166       subplot(224);
167       window3(min(muav),max(muav),roi,Pcurrent); title(['Updated image.
              Iteration = ' num2str(it)]);
168
169       subplot(221)
170       window3(0.13,0.35,roi,P);title (['Original Phantom']);
171       subplot(222)
172       plot(Pcurrent(135,:)); title(['Cupping']); axis('square');axis([0
              200 0.15 0.35]);
173
174       CF(it) = costfun(mu,Ie,d,SE)
175   end
176
177   its = [1:itmax];
178   figure;
179   plot(its,CF)
```

simdat.m

```
1   function RFs = simdat(mu,Ie,d,s)
2   % phi is the cost function that we want to minimize
3   % mu is the attenuation coef.
4   % I is the energy spectrum
5   % d is the relative density which models the small variation in
        attenuation
6   % within one material
7   global Pcurrent
8   %global RFsim
9   global RFpoly  % the measured polychromatic data
10  global Nmat    % number of materials
11  global Npic     % number of pixel
12  global pview    % # of views
13  global Q         % # of rays
14  %global Ie
15
16  %global D
```

```
17
18
19 %P =zeros(Npic,Npic); % instead of have a stack of images we use a
       single image with three different values (s==n) to represent the
       three materials
20
21 RP=zeros(Nmat,Q,pview);
22 for n = 1: Nmat  % this loop for every n materials
23     P =zeros(Npic,Npic); % instead of have a stack of images we use a
           single image with three different values (s==n) to represent the
            three materials
24     chi = (s==n); % find the pixels in the image P where the s =
           material n
25     P(chi) = d(chi); % this is the dj*snj
26     RP(n,:,:)=radon(P); %the radon data of the picture P(chi) which is
           dj*snj
27 end
28
29
30 RFs = zeros(size(RFpoly));% initialize a matrix with the simulated data
31
32
33 En = length(Ie);
34 for e = 1:En % outter loop for energy
35     tmp = zeros(size(RFpoly)); % temporary variable which represent the
            n sum
36     for n=1:Nmat % inner loop for the material
37         R1= zeros(size(RFpoly));
38         R1(:,:) = RP(n,:,:); % R1 is used instead of RP, to write the
               RP in term of 2 indesis instead of 3
39         tmp = tmp + mu(e,n)*R1; % the sum over n
40     end
41     RFs = RFs + Ie(e)*exp(-tmp); % to compute the e sum
42 end
43 RFs = -log(RFs); % the simulated polychromatic data
```

Appendix D

ISP Code

Source code for the ISP method.
All code provided by Maryam Alarfaj [1].

ISPmethod.m

```
% main routine for ISP method
global RFpoly  % the measured polychromatic data
global Nmat    % number of materials
global Npic    % number of pixel
global pview   % # of views
global Q       % total # of rays
global Ie
global mu
global d
global Pcurrent
global RFsim


itmax = 4; % number of iterations
Npic = 200;
[Q,pview] = size(radon(zeros(Npic,Npic)));
Nmat = 5; % number of materials

MX=200 ;  MY = 200; %matrix dimensions
roi=[-1  1 -1  1]; %roi=[xmin xmax ymin ymax]

circle = 1; % If circle = 1 image computed only inside

wmin = 0.14; wmax = 0.22;

% Specify parameters of ellipses for mathematical phantom.
```

```
% xe = vector of x-coordinates of centers
% ye = vector of y-coordinates of centers
% ae = vector of first half axes
% be = vector of second half axes
% alpha = vector of rotation angles (degrees)
%rho = vector of densities [mat1, mat2]
xe=[-0.24  0.24 ];
ye=[-0.24  0.24];
ae=[0.5  0.4];
be=[0.6  0.5];
alpha=[0  0];

Ie =[0.1; 0.3; 0.3; 0.2; 0.1];
alpha = alpha*pi/180;
rho =[0.999  -0.7340 ;...
     0.595  -0.3690 ;...
     0.416  -0.206;...
     0.265  -0.0820 ;...
     0.208  -0.0340 ];

rhosoft = rho(:,1) + rho(:,2);
rhodiff = -rho(:,2);
density = [rhosoft rhodiff/8 rhodiff rhodiff rhodiff/4].';

mu = zeros(length(Ie), Nmat);
mu(:,1) =0;
mu(:,2) =rhosoft;
mu(:,3) = rhosoft+rhodiff/8;
mu(:,4) = rhosoft+rhodiff/4;
mu(:,5) = rhosoft+rhodiff;

mu

Ne = length(Ie);
muav = mu(ceil(Ne/2),:)
mumax  = max(mu); mumax(mumax==0)=1;
Ellpar =[.9   .15   .15  .15   .15;...
          .9   .15   .15  .15   .15;...
            0  -.45  -.45  .45   .45;...
            0  -.45   .45  .45  -.45;...
            0     0     0    0     0].';


% ----------The measured polychromatic data RFpoly--------
pview = 180;
theta = [0:pview-1];
```

```
xp = [-143:143]/100;
RFpoly = zeros(length(xp),length(theta));
for j = 1:length(Ie)
    E = [density(:,j) Ellpar];
    RFpoly = RFpoly + Ie(j)* exp(-(Npic/2)*radon_ell(E,theta,xp));
end
RFpoly = -log(RFpoly); % the uncorrected polychromatic data
I = iradon(RFpoly,[0:179],'linear','Hamming',1,Npic);


%----------Computing the segmented Image----------
Pcurrent = I;
figure;
subplot(221)
window3(wmin,wmax,roi,Pcurrent);
 title(['Reconstruction from polychromatic data']);

 %----------find an initial threshold ----------
[t1,Ftmp,Dtmp,xdens,Nmat1]= findthreshold2(Pcurrent);

if Nmat1 < Nmat
    t = zeros(Nmat-1,1);
    t(1: Nmat1-2) = t1(1:Nmat1-2);
    t(Nmat1-1:Nmat-2) = t1(Nmat1-2)+ ([1:Nmat-Nmat1]/...
    (1+Nmat-Nmat1))*(t1(Nmat1-1)-t1(Nmat1-2));
    t(Nmat-1) = t1(Nmat1-1);
else
    t(1:Nmat-1) = t1(1:Nmat-1);
end
SE = threshold(Pcurrent,t);

d= ones(size(SE));

subplot(222);
window3(1,Nmat,roi,SE);
title(['Initial segmented image. Nmat = ' num2str(Nmat) ' materials.'])
    ;

C = costfun(mu,Ie,d,SE)

[SE1,phi1,t1] =optims3(t);

SE = SE1; t = t1;

subplot(223)
window3(1,Nmat,roi,SE);
```

```
118 title(['Segmented image with adjusted thresholds']);
119 RFsim = simdat(mu,Ie,d,SE);
120
121
122 %----------compute mu_refrence for ISP----------
123
124 [muISP1,B,V] = muISP(mu,Ie,SE);
125
126
127 %------ the simulated monochromatic data---------
128 RP=zeros(Nmat,Q,pview);
129 RFmono = zeros(size(RFpoly));
130 R2 = zeros(size(RFmono));
131 tmp1 = zeros(size(RFmono));
132 for n1= 1:Nmat
133     P =zeros(Npic,Npic);
134     chi = (SE==n1);
135     P(chi) = d(chi);
136     RP(n1,:,:)=radon(P);
137     R2(:,:) = RP(n1,:,:);
138     tmp1 = tmp1+ muISP1(n1)*R2;
139 end
140 RFmono = RFmono+tmp1;
141
142
143 %---------image update--------
144 RFcor = zeros(size(RFpoly));
145 RFcor = RFpoly+(RFmono-RFsim);
146 Pcurrent = iradon((RFcor),[0:179],'linear','Hamming',1,Npic);
147
148 max(max(RFcor))
149 min(min(RFcor))
150
151 subplot(224);
152 window3(min(muav),max(muav),roi,Pcurrent);
153 title(['Updated image. Iteration =1' ]);
154 CF = zeros(itmax,1);
155 CF(1) = costfun(mu,Ie,d,SE)
156
157 [t,Ftmp,Dtmp,xdens,Nmat1]= findthreshold2(Pcurrent);
158 if Nmat1 < Nmat
159     t = zeros(Nmat-1,1);
160     t(1: Nmat1-2) = t1(1:Nmat1-2);
161     t(Nmat1-1:Nmat-2) = t1(Nmat1-2)+ ([1:Nmat-Nmat1]/...
162     (1+Nmat-Nmat1))*(t1(Nmat1-1)-t1(Nmat1-2));
163     t(Nmat-1) = t1(Nmat1-1);
```

```
164 else
165     t(1:Nmat-1) = t1(1:Nmat-1);
166 end
167
168 CF = zeros(itmax,1);
169 for it = 1:itmax
170     it
171     [SE1,phi1,t1] =optims3(t);
172
173     SE = SE1; t = t1;
174     figure
175     subplot(223)
176     window3(1,Nmat,roi,SE);
177     title(['Segmented image with adjusted thresholds']);
178     RFsim = simdat(mu,Ie,d,SE);
179
180     [muISP1,B,V] = muISP(mu,Ie,SE);
181     RP=zeros(Nmat,Q,pview);
182     RFmono = zeros(size(RFpoly));
183     R2 = zeros(size(RFmono));
184     tmp1 = zeros(size(RFmono));
185
186     for n1= 1:Nmat
187         P =zeros(Npic,Npic);
188         chi = (SE==n1);
189         P(chi) = d(chi);
190         RP(n1,:,:)=radon(P);
191         R2(:,:) = RP(n1,:,:);
192         tmp1 = tmp1+ muISP1(n1)*R2;
193     end
194     RFmono = RFmono+tmp1;
195
196     RFcor = zeros(size(RFpoly));
197     RFcor = RFpoly+(RFmono-RFsim);
198     Pcurrent = iradon((RFcor),[0:179],'linear','Hamming',1,Npic);
199
200     max(max(RFcor))
201     min(min(RFcor))
202
203     subplot(224);
204     window3(min(muav),max(muav),roi,Pcurrent);
205      title(['Updated image. Iteration =1' ]);
206
207     subplot(224);
208     window3(min(muav),max(muav),roi,Pcurrent);
209     title(['Updated image. Iteration =' num2str(it)]);
```

```
210         CF(it) = costfun(mu,Ie,d,SE)
211 end
```

muISP.m

```
1  %--------------Computing the reference attenuation---------
2   function [muISP1,B,V] = muISP(mu,Ie,s)
3   % muISP is a function to compute the reference attenuation for ISP
4  global Pcurrent
5  global RFpoly  % the measured polychromatic data
6  global Nmat    % number of materials
7  global Npic    % number of pixel
8  global pview   % # of views
9  global Q       % # of rays
10 global d
11
12
13 d = ones(size(s));
14
15 RP=zeros(Nmat,Q,pview);
16 for n = 1: Nmat
17     P =zeros(Npic,Npic);
18     chi = (s==n);
19     P(chi) = d(chi);
20     RP(n,:,:)=radon(P);
21 end
22
23 V= zeros(Nmat,1);
24 RFs = zeros(size(RFpoly));
25
26 En = length(Ie);
27 for e = 1:En
28     tmp = zeros(size(RFpoly));
29     for n=1:Nmat
30         R1= zeros(size(RFpoly));
31         R1(:,:) = RP(n,:,:);
32         tmp = tmp + mu(e,n)*R1;
33     end
34     RFs = RFs + Ie(e)*exp(-tmp);
35 end
36
37 RFs = -log(RFs); % the simulated polychromatic data
38
39 for n= 1:Nmat
40     R1= zeros(size(RFpoly));
41         R1(:,:) = RP(n,:,:);
42         V(n) = sum(sum(R1.*RFs));
```

```
43 end
44
45  B = zeros(Nmat,Nmat);
46 for n = 1:Nmat
47       for m=1:Nmat
48       B(n,m) = sum(sum(RP(n,:,:).*RP(m,:,:)));
49       end
50 end
51
52 muISP1 = pinv(B)*V
```

Bibliography

[1] Maryam Khalid Alarfaj. A comparative study of two methods for the correction of beam hardening artifacts in x-ray computed tomography. 2012.

[2] Julia F Barrett and Nicholas Keat. Artifacts in ct: Recognition and avoidance. *Radiographics*, 24(6):1679–1691, 2004.

[3] Thorsten M Buzug. *Introduction to Computed Tomography: From Photon Statistics to Modern Cone-beam CT.* Springer, 2008.

[4] AJ Coleman and M Sinclair. A beam-hardening correction using dual-energy computed tomography. *Physics in Medicine and Biology*, 30(11):1251, 1985.

[5] Allan M Cormack and Godfrey N Hounsfield. The Nobel Prize in Physiology or Medicine 1979. *Nobelprize.org*, 2011.

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

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

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

[9] Gabor T Herman. *Image reconstruction from projections: Fundamentals of computerized tomography.* San Francisco : Academic Press, 1980.

[10] Gabor T Herman and Sushma S Trivedi. A comparative study of two postreconstruction beam hardening correction methods. *Medical Imaging, IEEE Transactions on*, 2(3):128–135, 1983.

[11] Hans Hornich. A tribute to Johann Radon. *Medical Imaging, IEEE Transactions on*, 5(4):169–169, 1986.

[12] Garth Kruger and Mirjan-Misko Nadrljanski. Computed tomography. *Radiopaedia.org*, 2013.

[13] Johan Nuyts, Sigrid Stroobants, Patrick Dupont, Stefaan Vleugels, Patrick Flamen, and Luc Mortelmans. Reducing loss of image quality because of the attenuation artifact in uncorrected pet whole-body images. *Journal of Nuclear Medicine*, 43(8):1054–1062, 2002.

[14] Elke Van de Casteele. Model-based approach for beam hardening correction and resolution measurements in microtomography. *Faculty Wetenschappen, Department Natuurkunde*, 2004.

[15] Gert Van Gompel, Katrien Van Slambrouck, Michel Defrise, K Joost Batenburg, Johan de Mey, Jan Sijbers, and Johan Nuyts. Iterative correction of beam hardening artifacts in ct. *Medical Physics*, 38:S36, 2011.

[16] Math Works. Inverse radon transform. *http://www.mathworks.com/help/images/ref/iradon.html*, 2013.