MATHEMATICAL PROBLEMS OF THERMOACOUSTIC AND COMPTON CAMERA IMAGING

A Dissertation

by

YULIA NEKOVA GEORGIEVA-HRISTOVA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2010

Major Subject: Mathematics

MATHEMATICAL PROBLEMS OF THERMOACOUSTIC AND COMPTON CAMERA IMAGING

A Dissertation

by

YULIA NEKOVA GEORGIEVA-HRISTOVA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Peter Kuchment
Committee Members,	Wolfgang Bangerth
	Jay R. Walton
	Mary McDougall
Head of Department,	Albert Boggess

August 2010

Major Subject: Mathematics

ABSTRACT

Mathematical Problems of Thermoacoustic
and Compton Camera Imaging. (August 2010)
Yulia Nekova Georgieva-Hristova, B.S., Sofia University
Chair of Advisory Committee: Dr. Peter Kuchment

The results presented in this dissertation concern two different types of tomographic imaging. The first part of the dissertation is devoted to the time reversal method for approximate reconstruction of images in thermoacoustic tomography. A thorough numerical study of the method is presented. Error estimates of the time reversal approximation are provided. In the second part of the dissertation a type of emission tomography, called Compton camera imaging is considered. The mathematical problem arising in Compton camera imaging is the inversion of the cone transform. We present three methods for inversion of this transform in two dimensions. Numerical examples of reconstructions by these methods are also provided. Lastly, we turn to a problem of significance in homeland security, namely the detection of geometrically small, low emission sources in the presence of a large background radiation. We consider the use of Compton type detectors for this purpose and describe an efficient method for detection of such sources. Numerical examples demonstrating this method are also provided.

To my father, Neko Nikiforov Georgiev

ACKNOWLEDGMENTS

I would like express my deepest gratitude to Prof. Peter Kuchment not only for introducing me to the interesting subjects of my dissertation and for his outstanding teaching, but also for his patience, encouragement and belief in me. I thank him for his invaluable advice on numerous topics - from research and teaching, through job search and writing style. I deeply appreciate his giving me and pointing out to me numerous opportunities for professional development, for thoroughly reading the many versions of this dissertation, and for always looking out for me.

I am grateful to Prof. Jay R. Walton for his mentorship during my first years as a graduate student. He taught me a lot about several exciting topics in mathematical modeling and his excellent course in continuum mechanics provided me with mathematical techniques and the theoretical background needed in my current, and I am sure, future research endeavors.

I would like to thank Prof. Bangerth for patiently teaching me to program in C^{++} and to use deal.ii library. I am also grateful to him for the interesting discussions of thermoacoustic tomography. His lectures on numerical inversion techniques for differential equations were a great introduction to the subject.

I would like to thank Prof. Mary McDougall not only for serving as a member of my graduate committee, but also for bringing my attention to an interesting paper by Gordon and Herman [1].

I would also like to thank Prof. Leonid Kunyansky for providing some of his excellent source codes during the Summer Graduate Workshop on Inverse Problems,

2009 in MSRI, from which I learned how to effectively structure and organize my own programs.

I am grateful to all my teachers at Texas A&M University, Politechnico di Torino and Sofia University for sharing their knowledge with me.

I thank my family and parents for their encouragement and support throughout the years. I am especially grateful to my husband, Aleksandar, and to our wonderful son Pavel, for giving me strength in difficult times and for bringing me so much happiness.

I am indebted to all my friends in College Station for always being there for me and for selflessly helping me and my family whenever we needed.

Last but not least, I would like to thank the Department of Mathematics for giving me the opportunity to pursue my doctoral degree at Texas A&M University. I am very appreciative of the financial support provided by the Department of Mathematics and the National Science Foundation through grants DMS 0715090, DMS 0604778 and DMS 0908208.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Computerized tomography is a term which refers to technologies used for noninvasive imaging of interiors of non-transparent objects, with applications in medical imaging, industrial non-destructive testing, geophysics, astrophysics and homeland security. All types of computerized tomography have one feature in common, namely, information about the imaged object is obtained by measurements outside the object and then mathematics is used to compute an image of the interior of the object. Typically, different types of tomography pose different mathematical problems. For example, one of the most important and well studied tomographic techniques is the X-ray CT, which is mostly used in medicine. In order to reconstruct an image of the interior of an object, one has to invert the Radon transform. We give some of the basic properties of this transform in Section 1.1, as we will need them later.

In this dissertation two tomographic techniques are considered. Chapter II is devoted to thermoacoustic tomography (TAT), which is a realtively new medical imaging technique, aimed at detection of early stages of breast cancer. Thermoacoustic tomography achieves high contrast and high resolution images, while being safe and inexpensive. The mathematical problem which arises in TAT is the inverse observation problem for the wave equation. A closely related imaging method, called optoacoustic tomography (OAT), leads to the same mathematical problem. Often

---

This dissertation follows the style of the *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control.*

the term photocaoustic tomography (PAT) is used for both OAT and TAT. In this dissertation we will refer to TAT, although the presented results hold also for OAT. Surveys of PAT imaging and the mathematical problems arising in it can be found in [2]–[9].

The second part of this dissertation considers a problem from emission tomography, called Compton camera imaging. In emission tomography, particles emitted from the inside of an object are detected on the boundary of the object. Then, one attempts to find the distribution of sources inside the object. Compton cameras are a type of γ-ray detectors that find application in fields like nuclear medicine, astrophysics and monitoring nuclear power plants. The mathematical problem associated with Compton camera imaging is that of inverting the so-called the cone transform. This integral transform has been studied recently, however many questions remain open. References to papers on Compton camera imaging can be found in Chapter III. In the first part of this chapter some results regarding the two-dimensional cone transform are presented. In the second part, we turn to the problem of detecting illicit nuclear material and the use of Compton type detectors and the results from the first part for this purpose.

The dissertation is organized as follows.

In Section 1.1 we define several integral transforms that are used in the later chapters. We also give some of the basic properties of these transforms.

Chapter II is devoted to thermoacoustic tomography (TAT). Section 2.1 contains a brief introduction to thermoacoustic tomography and some of the available reconstruction techniques. The time reversal method for reconstruction in TAT is

described in detail in Section 2.2, and results of its numerical implementation are provided and discussed in Section 2.3. An error estimate for the time reversal approximation is presented in Section 2.4 and numerical simulations confirming the results from this section can be found in Section 2.5. Section 2.6 contains some concluding remarks. The Matlab codes used for the numerical experiments discussed in this chapter are provided in Appendix A.

In Chapter III we consider problems related to Compton camera imaging. Section 3.1 contains an introduction to Compton camera imaging and a brief description of the available mathematical results. In Section 3.2 the two-dimensional cone transform is studied and numerical reconstructions are presented in Section 3.3. In Section 3.4 we consider the problem of detection of low emission geometrically small sources and Section 3.5 contains the results of our numerical simulations. Concluding remarks can be found in Section 3.6. The Matlab codes used for the computations discussed in this chapter are provided in Appendix B.

1.1. Definitions and Properties of Some Integral Transforms

In this section we give some definitions and properties of integral transforms which we be needed later. More details can be found, for example, in [10] and [11].

Let $f(x)$ belong to the Schwartz space $\mathcal{S}(\mathbb{R}^n)$. The *Fourier Transform* of f is defined as

$$\hat{f}(\xi) := \mathcal{F}f(\xi) := (2\pi)^{-n/2} \int_{\mathbb{R}^n} f(x) e^{-ix\cdot\xi}\, dx, \quad \xi \in \mathbb{R}^n \tag{1.1}$$

The inverse Fourier transform is

$$f(x) = \mathcal{F}^{-1}\hat{f}(x) = (2\pi)^{-n/2} \int_{\mathbb{R}^n} f(x) e^{ix\cdot\xi}\, dx \tag{1.2}$$

The Fourier transform can be extended as a bijection from $L^2(\mathbb{R}^n)$ to itsself.

The *Hilbert transform* of a function u is defined by the following principal value integral (when it exists)

$$Hu(t) = p.v.\frac{1}{\pi}\int_{-\infty}^{\infty} \frac{u(s)}{t-s}\, ds = \lim_{\varepsilon\to 0}\frac{1}{\pi}\left(\int_{-\infty}^{t-\varepsilon} \frac{u(s)}{t-s}\, ds + \int_{t+\varepsilon}^{\infty} \frac{u(s)}{t-s}\, ds,\right) \tag{1.3}$$

which can also be written as

$$Hu(t) = \lim_{\varepsilon\to 0^+}\frac{1}{\pi}\int_{\varepsilon}^{\infty} \frac{u(t-s)-u(t+s)}{s}\, ds. \tag{1.4}$$

If u is α-Hölder continuous and with compact support, then $Hu(t)$ is a also Hölder continuous (e.g. [12]) and we write

$$Hu(t) = \frac{1}{\pi}\int_{0}^{\infty} \frac{u(t-s)-u(t+s)}{s}\, ds. \tag{1.5}$$

The Hilbert transform has the following properties:

1. $H^2 f(s) = -f(s)$

2. $\mathcal{F}(Hf)(\sigma) = -i\ \mathrm{sgn}(\sigma)\mathcal{F}f(\sigma)$

One of the most important integral transforms used in tomography is the n-dimensional *Radon transform*, which we will denote by $\mathcal{R} : L_2(\mathbb{R}^n) \to L_2(S^{n-1}\times\mathbb{R})$. The Radon transform maps each function $f(x) \in L^2(\mathbb{R}^n)$ into the the set of its

integrals over hyperplanes, i.e.

$$\mathcal{R}f(\omega, s) = \int\limits_{x\cdot\omega=s} f(x)\, dx = \int\limits_{\omega^\perp} f(s\omega + y)\, dy \tag{1.6}$$

Here $\omega \in S^{n-1}$, $s \in \mathbb{R}$ and $\omega^\perp$ denotes the subspace perpendicular to ω. The set $L_{(\omega,s)} := \{x \in \mathbb{R}^n | x \cdot \omega = s\}$ is the hyperplane perpendicular to ω, which lies at distance s from the origin.

We will mostly use the Radon transform in two dimensions, which maps a function $f(x),\ x \in \mathbb{R}^2$, into the set of its line integrals, i.e.

$$\mathcal{R}f(\omega, s) = \int\limits_{x\cdot\omega=s} f(x)\, dx = \int_{-\infty}^{\infty} f(s\omega + t\omega^\perp)\, dt, \tag{1.7}$$

where $\omega^\perp$ denotes the unit vector perpendicular to ω. Then, $L_{(\omega,s)}$ is the line perpendicular to ω which lies at distance s from the origin, see Fig. 1.1.

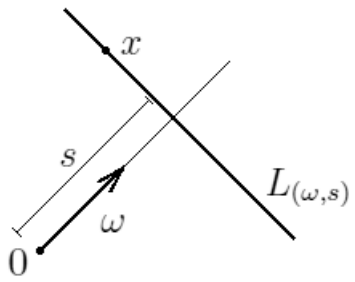

Fig. 1.1. The line $L_{(\omega,s)}$ is defined by the unit vector ω perpendicular to it, and by the distance s from the origin.

The *divergent beam transform* (in 2D also *fanbeam transform*) is defined by

$$\mathcal{D}f(a,\omega) = \int_0^\infty f(a+t\omega)\, dt. \tag{1.8}$$

Thus, $\mathcal{D}f(a,\omega)$ is the integral of f along the half-line starting at the point $a \in \mathbb{R}^n$ in the direction of $\omega \in S^{n-1}$.

The Radon transform has the following properties:

1. The function $\mathcal{R}f(\omega, s)$ is even, i.e. $\mathcal{R}f(-\omega,-s) = \mathcal{R}(\omega,s)$.

2. Denote the unit ball in $\mathbb{R}^n$ by B^n. Then

$$\mathcal{R} : H_0^\alpha(B^n) \to H^{\alpha+(n-1)/2}(S^{n-1}\times\mathbb{R}). \tag{1.9}$$

3. The dual to $\mathcal{R}$ operator is called the *backprojection* operator and is given by

$$\mathcal{R}^\# g(x) := \int_{S^{n-1}} g(\omega, x\cdot\omega)\, d\omega \tag{1.10}$$

   Here $g(\omega,s) \in L_1(S^{n-1}\times\mathbb{R})$.

4. Consider the backprojection operator, applied to $\mathcal{R}f$

$$\mathcal{R}^\#\mathcal{R}f(x) := \int_{S^{n-1}} \int_{L_{(\omega,x\cdot\omega)}} f(y)\, dy d\omega. \tag{1.11}$$

   We can say that, for every x, $\mathcal{R}^\#\mathcal{R}f(x)$ is the "sum" all line integrals of f which pass through the point x. We will refer to $\mathcal{R}^\#\mathcal{R}f(x)$ as backprojection reconstruction.

5. The following relation holds

$$\mathcal{R}^{\#}\mathcal{R}f = |S^{n-2}|\,\frac{1}{x} * f, \tag{1.12}$$

where $|S^{n-2}|$ is the surface of the unit sphere in $\mathbb{R}^{n-1}$.

6. The *Fourier Slice Formula* gives a connection between Radon and Fourier transforms.

$$\mathcal{F}_1\mathcal{R}f(\omega,\sigma) = (2\pi)^{(n-1)/2}\hat{f}(\sigma\omega), \quad \sigma \in \mathbb{R}, \tag{1.13}$$

where $\mathcal{F}_1$ denotes the one-dimensional Fourier transform with respect to s. In other words, the (n-dimensional) Fourier transform is the Fourier transform with respect to the linear variable of the Radon transform. The Fourier Slice Formula provides an inversion formula for the Radon transform. Indeed, one only has to apply the inverse Fourier transform to (1.13) in order to recover f.

7. The *Filtered Backprojection (FBP) Formula* is a widely used formula for inversion of the Radon transform. In 3D it reads

$$f(x) = \frac{1}{4\pi^2}\mathcal{R}^{\#}\left(\frac{\partial^2}{\partial s^2}\mathcal{R}f\right)(x) \tag{1.14}$$

In 2D FBP takes the form

$$f(x) = \frac{1}{4\pi}\mathcal{R}^{\#}\left(H\frac{\partial}{\partial s}\mathcal{R}f\right)(x), \tag{1.15}$$

where the Hilbert transform is taken with respect to s.

CHAPTER II

TIME REVERSAL IN THERMOACOUSTIC TOMOGRAPHY

2.1. Introduction to Thermoacoustic Tomography

[1] Thermoacoustic tomography (TAT) is a hybrid medical imaging technique that utilizes the electromagnetic energy absorption properties of tissue and ultrasound propagation. TAT images are characterized by high resolution and high contrast between cancerous and noncancerous tissues. Other advantages of TAT are safety and low cost. The main application of thermoacoustic tomography is in detection of early stages of breast cancer. A more detailed description of thermoacoustic imaging and a mathematical model follows. The first step in TAT is the irradiation of an object of interest with a short electromagnetic pulse. A part of the electromagnetic energy is absorbed throughout the tissue, which causes a thermoelastic expansion. This leads to a pressure wave propagating through the object. The pressure is measured by ultrasound transducers located on an *observation surface* S surrounding the object, see Fig. 2.1. The collected information is then used to reconstruct the initial pressure at every point inside the object. Cancerous tissue absorbs significantly more microwave energy than normal tissue [15], and it is known that the initial pressure is roughly proportional to the amount of absorbed electromagnetic energy [2], [16], [17]. This leads to a large contrast between cancerous and normal tissues in TAT images [15]. On the other hand, the submillimeter resolution of TAT images is

[1]Some parts from this chapter were reproduced from [13], [14] with permission from IOP Publishing Limited.

due to ultrasound [2], [15],[18].

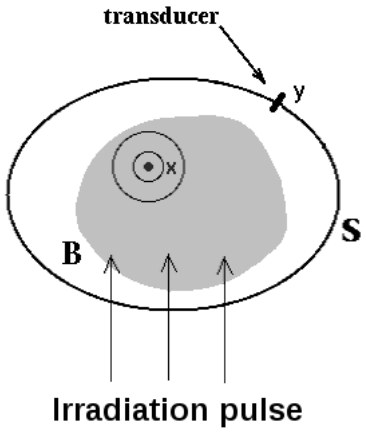

Fig. 2.1. Schematic representation of a TAT scanning system.

2.1.1. Mathematical Model of TAT

The equations describing thermoacoustic wave generation have been discussed in detail in the literature. We give a brief derivation of the model here, and refer to reader to [3, Ch. 12] and [16] for further details.

Let $p(x,t)$ denote the acoustic pressure change at location $x \in \mathbb{R}^3$ and time t, and let $u(x,t)$ be the medium displacement. We say that thermal confinement holds, when the heating pulse is shorter than the time of dissipation of the absorbed energy. Under thermal confinement the temperature rise $T(x,t)$ satisfies the equation:

$$\rho C_V \frac{\partial T(x,t)}{\partial t} = H(x,t). \tag{2.1}$$

Here ρ is the mass density, C_V is the specific heat capacity at constant volume, and

$H(x,t)$ is the heating function defined as thermal energy converted per unit volume and unit time.

We assume that the mass density, $\rho(x)$, varies slowly, i.e.

$$\rho = \rho_0(1+s),\ s \ll 1, \tag{2.2}$$

where ρ_0 is the equilibrium mass density (for soft tissue $\rho_0 \approx\ 1000kg/m^3$).

The thermal expansion equation (generalized Hooke's law) reads as

$$\operatorname{div} u(x,t) = -\kappa(x)p(x,t) + \beta T(x,t) \tag{2.3}$$

Here κ is the isothermal compressibility and β denotes the thermal coefficient of volume expansion.

The linear inviscid force equation (the equation of motion), under the assumption (2.2), reads

$$\rho_0 \frac{\partial^2}{\partial t^2} u(x,t) = -\nabla p(x,t) \tag{2.4}$$

Taking divergence of the last equation leads to

$$\rho_0 \frac{\partial^2}{\partial t^2}(\operatorname{div} u) = -\Delta p$$

Substituting (2.3) into the above equation results in

$$\frac{\partial^2}{\partial t^2} p(x,t) - c^2(x)\Delta p = \frac{\beta}{\kappa(x)} \frac{\partial^2 T(x,t)}{\partial t^2} \tag{2.5}$$

Here $c^2(x) = 1/\kappa(x)\rho_0$ is the speed of sound in the imaged tissue. Relation (2.1) implies

$$\frac{\partial^2}{\partial t^2} p(x,t) - c^2(x)\Delta p = \frac{\beta}{C_P} c^2(x) \frac{\partial H}{\partial t} \tag{2.6}$$

Since the irradiation pulse is short, we can assume that $H(x,t) = I(x)\delta(t)$, where $I(x)$ is the radiation intensity and δ is the delta function. Let us denote $f(x) = \frac{\beta}{C_P}c^2(x)I(x)$. Then the equation for the pressure $p(x,t)$ is

$$\begin{cases} p_{tt} = c^2(x)\Delta p, & t \geq 0, \quad x \in \mathbb{R}^n \\ p(x,0) = f(x), & p_t(x,0) = 0, \\ p(y,t) = g(y,t) & \text{for } y \in S, \quad t \geq 0. \end{cases} \tag{2.7}$$

Here S is the observation surface surrounding the object and B be the domain bounded by S. The function $g(y,t)$ is the value of the pressure at time t measured at transducer's location $y \in S$, and $f(x)$ is the initial pressure, which we seek to find. The function $f(x)$ is assumed to have a compact support in B. One thus faces the problem of inverting the mapping $R\colon f \to g$ from the initial pressure to the measured data. It must be noted that a similar imaging modality, called optoacoustic tomography, exists, in which the thermoelastic expansion is triggered by a laser. The mathematical model for this type of tomography is the same as for TAT.

2.1.2. Reconstruction Methods in TAT

Here we describe briefly the main types of reconstruction procedures used in TAT. One can find detailed surveys on this topic in [4]–[9].

2.1.2.1. Filtered Backprojection Formulas

In the case when the sound speed is constant, i.e. $c(x) \equiv c$, the wave equation 2.7 can be solved using the well known Poisson-Kirchhoff formulas (e.g. [19]):

$$p(x,t) = c_n \left[\left(\frac{\partial}{\partial t} \frac{1}{t} \right)^{\frac{n-1}{2}} Rf \right] (x, ct), \quad x \in \mathbb{R}^n, \quad t > 0 \tag{2.8}$$

where $c_n > 0$ is a constant and

$$(Rf)(x,t) = \int_{|y|=1} f(x+ty)\, dA(y) \tag{2.9}$$

is the spherical mean of f on the sphere centered at x with radius t, dA is the surface area measure on this sphere. Thus, the knowledge of the values of the pressure on the observation surface S implies the knowledge of the values of the spherical mean transform of f, $Rf(x,t)$, for points $x \in S$. The aim now is to invert R_S - the spherical mean operator with centers on S.

The filtered backprojection formulas are closed form formulas which provide inversions of R_S. For the case when S is a sphere with radius r, such formulas were provided in [20]–[22]. One of these formulas in three dimensions reads as follows:

$$f(x) = -\frac{1}{8\pi^2 r} \int_S \left(\frac{d^2}{dt^2} R_S f(y,t) \right) \Big|_{t=|y-x|} dA(y)$$

Recently, a family of filtered backprojection formulas, unifying all such previously known formulas, was given in [23]. The filtered backprojection formulas are fast and accurate, however the only available formulas are for spherical observation surfaces and constant sound speed. Moreover, if the support of f has parts outside the domain surrounded by S, these formulas reconstruct f inside S incorrectly [5],[6].

2.1.2.2. Eigenfunction Expansion Methods

The eigenfunction expansion methods work for arbitrary closed observation surfaces S and initial pressures f with bounded support not necessarily inside S. The only significant condition on the sound speed is that $c(x)$ is non-trapping. This condition will be discussed later in Section 2.2. In [24] the following, very general, reconstruction formula was provided:

$$f(x) = \sum_k f_k \psi_k(x),$$

where

$$f_k = -\lambda_k^{-1} \int_0^\infty \sin(\lambda_k t) g_k(t)\, dt\ ;\ g_k(t) = \int_S g(x,t) \overline{\frac{\partial \psi_k}{\partial \nu}(x)}\, dx$$

$\psi_k(x), \lambda_k^2$ are the eigenfunctions and eigenvalues of $-c^2(x)\Delta$ in the interior of S with zero Dirichlet conditions on S. This method was shown to be very fast and precise when the sound speed is constant and the observation surface is a cube [25]. However, if the sound speed is variable, the method is computationally very expensive, as a large number of eigenvalues and eigenfunctions of $-c^2(x)\Delta$ need to be computed numerically.

2.1.2.3. Time Reversal Method

In essence, the time reversal method works by solving the wave equation inside B in reversed time order, using the detected signals at S as boundary conditions. At time $t = 0$, the solution approximates the initial pressure $f(x)$ we are looking

for. Time reversal can be used for approximation of the initial pressure for a broad range of sound speeds and arbitrary observation surfaces [13], [14], [26], [27]. The method does not require the support of f to be constrained to the interior of the imaged region. Moreover, time reversal is easy to implement numerically [13], [14], [26]. All of these characteristics make time reversal the most versatile among the reconstruction methods in TAT.

The time reversal method is the object of study in the first part of this dissertation. It should be noted that, although we are interested in applications to thermoacoustic tomography, the time reversal method for the wave equation has been used in other applications [28]–[30]. Our results apply in all of these cases.

2.1.2.4. Algebraic Reconstruction Techniques

Algebraic reconstruction techniques are yet another option for reconstruction in TAT. These methods allow for quality reconstructions when the observation surface is open [31] and can be used in cases of heterogeneous acoustic media [32]. However, these are iterative algorithms and as such are computationally expensive. Faster converging algorithms were developed in [33], [34], where approximate analytic formulas were used as a preconditioner for iterative solvers.

The remainder of this chapter is organized as follows. In Section 2.2 a non-trapping condition on the sound speed is recalled and the time reversal method is described. Numerical reconstructions based on time reversal are presented and dis-

cussed in Section 2.3. In Section 2.4 an error estimate for the time-reversal approximation is proved. Numerical study of errors is provided in Section 2.5. Section 2.6 contains some concluding remarks.

2.2. Non-trapping Condition and the Time Reversal Method

In this section we recall the notion of a non-trapping sound speed and we give a more detailed description of the time reversal method.

We will be interested in the initial value problem:

$$\begin{cases} p_{tt} = c^2(x)\Delta p, \quad t \geq 0, \quad x \in \mathbb{R}^n \\ p(x,0) = f_1(x), \quad p_t(x,0) = f_2(x), \end{cases} \tag{2.10}$$

where $c(x) > c > 0$, $c(x) \in C^\infty(\mathbb{R}^n)$, and $c(x) - 1$, as well as $f_1(x)$ and $f_2(x)$, has compact support. Consider the following Hamiltonian system in $2n$ real variables (x, ξ) with Hamiltonian $H = \frac{c^2(x)}{2}|\xi|^2$:

$$\begin{cases} x_t' = \frac{\partial H}{\partial \xi} = c^2(x)\xi \\ \xi_t' = -\frac{\partial H}{\partial x} = -\frac{1}{2}\nabla\left(c^2(x)\right)|\xi|^2 \\ x|_{t=0} = x_0, \xi|_{t=0} = \xi_0. \end{cases} \tag{2.11}$$

The solutions of this system are called *bicharacteristics* and their projections into the x-space $\mathbb{R}^n_x$ are called *rays*.

Definition 1. *We say that the non-trapping condition holds, if all rays (with $\xi_0 \neq 0$) tend to infinity when $t \to \infty$.*

Let us assume that the speed $c(x)$ of ultrasound in the tissue is smooth and strictly positive $c(x) > c > 0$, and that $c(x) \equiv 1$ for large values of $|x|$. It is known that the singularities of the solution of (2.10) propagate along bicharacteristics (e.g. [35]–[37]). Therefore, due to the non-trapping condition imposed on $c(x)$, it follows that for any distributions f_1, f_2 with compact supports, the singularities of the

solution move away to infinity as $t \to \infty$. Then, for a sufficiently large t, the solution $p(x,t)$ is infinitely differentiable inside any given bounded domain B. Moreover, the following estimates hold:

Theorem 2. *(e.g. [38] ,[37]) Under the conditions imposed on the sound speed $c(x)$ and $f(x)$ and for any bounded domain B, the solution of (2.10) satisfies the estimates*

$$\left| \frac{\partial^{k+|m|} p}{\partial_t^k \partial_x^m} \right| \leq C\eta_k(t) \left(\|f_1\|_{L^2} + \|f_2\|_{L^2} \right), \ x \in B, \ t > T_0, \tag{2.12}$$

for any multi-index (k,m), where $\eta_k(t) = t^{-n+1-k}$ for even n, and $\eta_k(t) = e^{-\delta t}$ for odd n. Here δ is a positive constant depending only on $c(x)$, T_0 depends on the domain B, and the C depends on B and the multi-index (k,m).

We will make use of the above theorem in estimating the error of time reversal reconstructions. Note that in the case of trapping sound speed the local energy of the solution of (2.10) still decreases in any compact domain, but, in general, there is no uniform local energy decay estimate [39], [40].

We now turn to describing the time-reversal method in the way it is applied to the thermoacoustic tomography problem.

The Huygens' principle states that in odd dimensions and when the sound speed is constant, for any initial source with bounded support and for any bounded domain, there is a moment in time when the wave leaves the domain (e.g. [41]). Thus, given an initial pressure $f(x)$ with a bounded support, there is a time T when the wave inside the domain B, bounded by the observation surface S, vanishes for all $t \geq T$. Then, to reconstruct $f(x)$ we can simply "rewind" the solution, i.e. we can solve the wave equation backwards in time inside B with zero initial conditions at $t = T$ and

boundary conditions given by the measured data on S. At time $t = 0$ the solution of this problem will be equal to $f(x)$.

In even dimensions or when the sound speed is variable, the Huygens' principle does not hold anymore. Nevertheless, we could still try to backpropagate the wave, hoping to approximate the initial pressure. This is the main idea of the *time reversal method* and we will now give a precise description of the method in the general case of a variable sound speed and dimension n.

To reconstruct the initial pressure $f(x)$ inside B, we try to reverse the time (starting from time $t = T$ and going back to $t = 0$) and solve the following problem:

$$\begin{cases} u_{tt} = c^2(x)\Delta u & \text{in } B \times [0,T] \\ u(x,T) = p(x,T) & \\ u_t(x,T) = p_t(x,T) & \\ u|_S(x,t) = g(x,t) & \text{on } S \times [0,T], \end{cases} \tag{2.13}$$

where $g = p|_{S\times[0,T]}$ is the restriction of p onto $S \times [0,T]$. At time $t = 0$ the solution of this problem equals $f(x)$[2]. Obviously, this reconstruction requires the knowledge of the solution to (2.7) on the cylinder $S \times [0,T]$ and inside B at time T. The values of p on the cylinder can be obtained from the measurements of the transducers placed on S. However, the values of the pressure inside B at time T are not known. Nevertheless, due to Theorem 2, we do know that the solution inside B decays with time. Therefore, after some time $T > T_0$ it is reasonable to approximate $p(x,T)$ and

[2]Note that $u(x,t)$ is simply the restriction of $p(x,t)$ onto $B \times [0,T]$.

$p_t(x,T)$ with zero. This is how we arrive to the approximate reconstruction problem:

$$\begin{cases} v_{tt} = c^2(x)\Delta v & \text{in } B \times [0,T] \\ v(x,T) = 0, \quad v_t(x,T) = 0 & \\ v|_S(x,t) = g(x,t)\varphi_\varepsilon(t) & \text{on } S \times [0,T]. \end{cases} \tag{2.14}$$

Here $\varphi_\varepsilon(t)$ is a smooth cut-off function that equals to 1 in $(-\infty, T-\varepsilon]$ (where $T-\varepsilon > T_0$) and vanishes for $t \geq T$. As it will be explained below, at time $t=0$ the solution to (2.14) approximates $f(x)$. In Section 2.4 we estimate the decay of this error with respect to the cut-off time T in the case of a non-trapping sound speed.

Another variation of the time reversal procedure exists in the literature [27]. In this version, the analogue of (2.14) is

$$\begin{cases} v_{tt} = c^2(x)\Delta v & \text{in } B \times [0,T] \\ v(x,T) = \phi(x), \quad v_t(x,T) = 0 & \\ v|_S(x,t) = g(x,t) & \text{on } S \times [0,T], \end{cases}$$

where $\phi(x)$ is the solution of

$$\Delta\phi = 0 \text{ in } B, \quad \phi|_S = g(\cdot, T).$$

2.3. Numerical Reconstructions Using Time Reversal

In this section we present 2D time reversal reconstructions from numerically generated data. Reconstructions for various sound speeds are shown. In all examples below the observation surface S is the unit sphere and we reconstruct the initial pressure inside the unit disk B. For our computations we used the rectilinear finite

difference scheme, implemented in Matlab. For the simulation of phantom data the following problem was solved

$$\begin{cases} p_{tt} - c_0^2 \Delta p = 0 \qquad \text{in} \quad D \times (0, T_0] \\ p(x,0) = f(x)\ p_t(x,0) = 0 \\ p|_{\partial D} = 0, \end{cases}$$

where $D = [-a, a]^2$ was a square containing S and large enough to ensure that no reflections off its boundary would reach S for time T_0. The values of the solution on S were recorded for all time steps. The time T_0, the domain D and the space and time step-sizes were adjusted depending on the situation. For the reconstruction part, the same equation and difference scheme were used (but with a different mesh, to avoid committing inverse crime), this time on the square $[-1.2, 1.2]^2$. On each time step the prerecorded values of the forward wave on S were enforced on the solution of the reconstruction problem. More details regarding the implementation, as well as the actual programming codes, can be found in Appendix A.

2.3.1. Constant Sound Speed

We begin with the most favorable situation: a constant speed that we choose to be equal to 1. Thus, it takes time $t = 2$ for the front of a wave moving with unit speed to cross the disk B. Although the sound speed is constant, Huygens' principle does not hold in $2D$, so the wave remains in B indefinitely. A numerical phantom consisting of two round objects is shown in Fig. 2.2. On the same figure a couple of time reversal reconstructions are shown. For the first reconstruction a cut-off time $T = 2$ was used, while for the second we waited longer, until time $T = 3$, before

initiating the reconstruction. We observe that taking larger cut-off times improves the result, as is expected. In Fig. 2.3 one finds a reconstruction of the Shepp-Logan phantom.

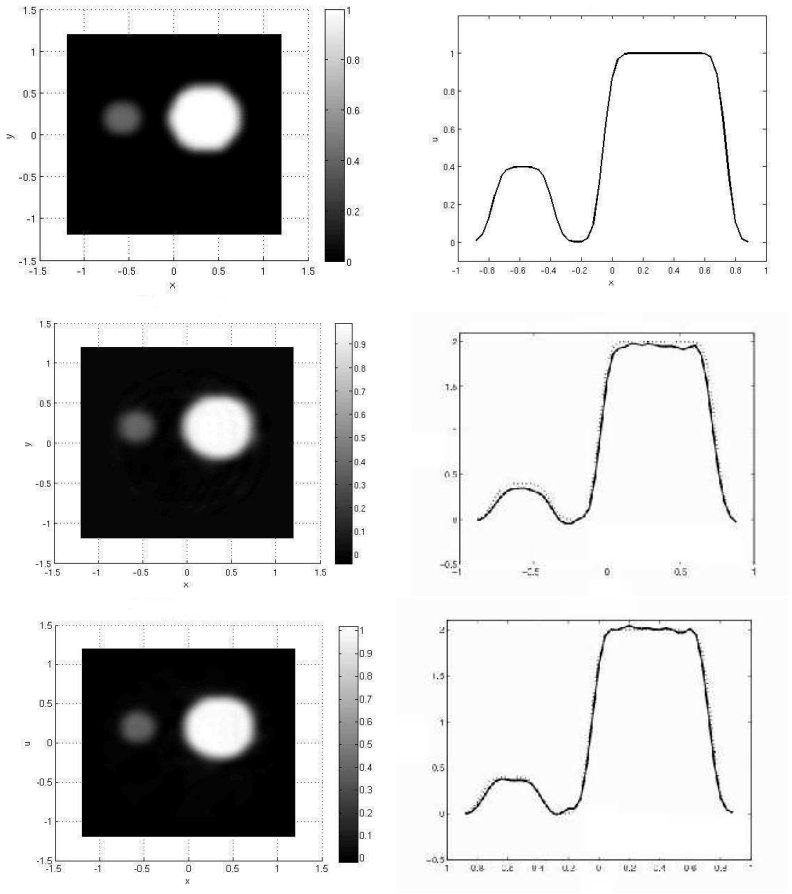

Fig. 2.2. The phantom and its axial profile (top), reconstruction for $c = 1$ and $T = 2$ (center), reconstruction for $c = 1$ and $T = 3$ (bottom).

These examples show that the time reversal method works well with a constant sound speed in $2D$, in spite of it being theoretically not exact and given the slow

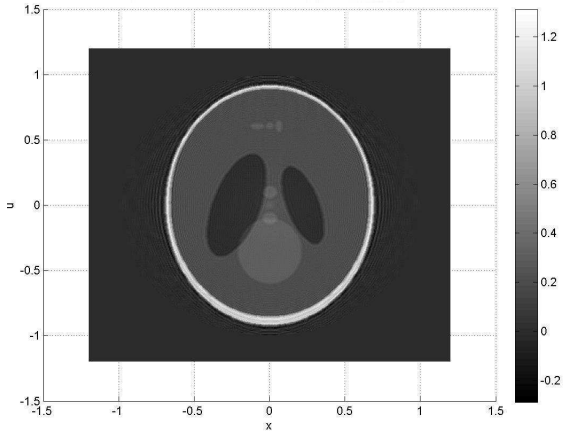

Fig. 2.3. Reconstruction for $c = 1$ and $T = 3$ of the Shepp-Logan phantom.

energy decay (see (2)).

Let us now illustrate some features of time reversal discussed before that are not available with the known backprojection formulas.

First of all, presence of a part of the function $f(x)$ outside the observation surface S does not hinder the reconstruction of f *inside* S. Fig. 2.4 shows a phantom that has a part of it outside the unit circle S and its reconstruction inside S. Clearly, there are no artifacts, which one would get with the analytic inversion formulas [5], [6].

Another interesting feature is that the initial velocity in (2.7) being zero (which is assumed in all analytic inversion formulas) is not necessary. We will show a corresponding example in the next section, with a variable sound speed. It works even better with a constant speed.

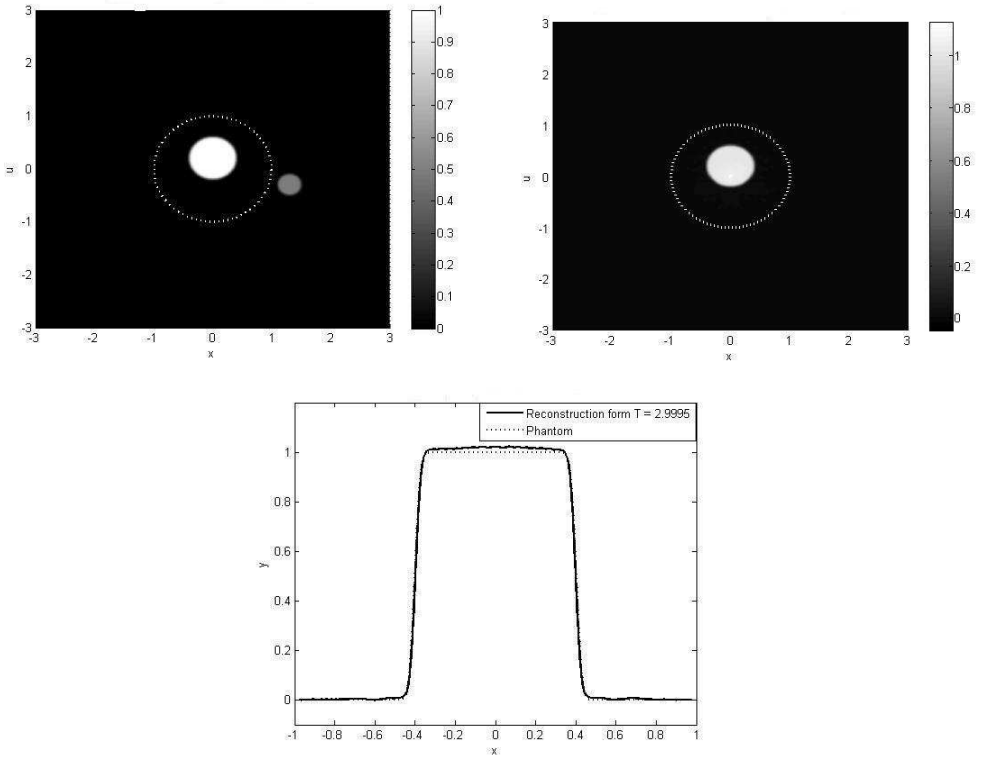

Fig. 2.4. Phantom (top left) with a part outside the observation surface and its reconstruction (top right) and comparison with the original (bttom).

2.3.2. Non-trapping Variable Speed

Let us now experiment with the time reversal method for a (known) variable speed under the non-trapping condition. In Fig. 2.5 one can see the speed profile and the phantom that were used in calculations, as well as the corresponding reconstructions. We now present an example where the initial velocity is non-zero. Fig. 2.6 shows the sound speed, the initial velocity and the initial value phantom (i.e., $f(x)$ to be recon-

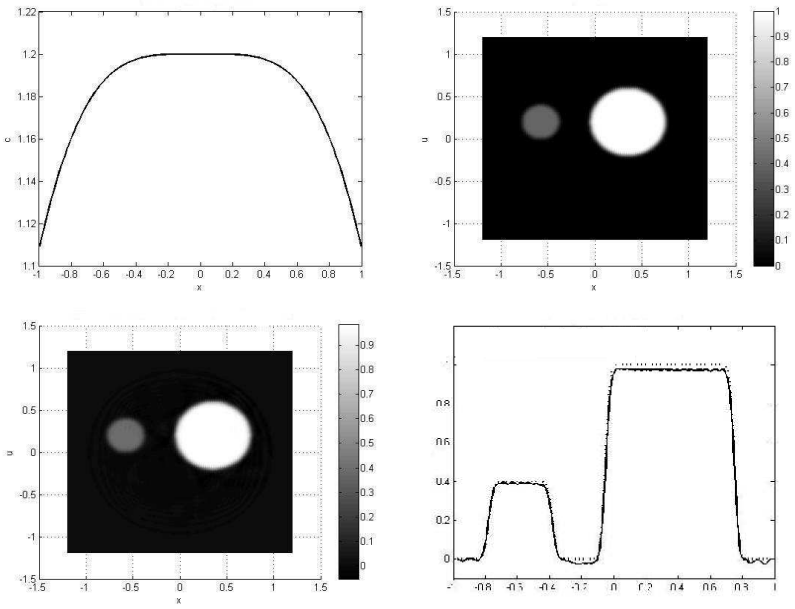

Fig. 2.5. A non-trapping (radial) speed profile (top left) and a phantom (top right). Reconstruction from time $T = 1.7$ (bottom left) and a profile plot of the phantom and the reconstruction.

structed). The reconstruction of the phantom shown in Fig. 2.6 confirms that the presence of a non-zero initial velocity does not hamper the reconstruction. Moreover, although this is not of interest for TAT, the initial velocity can also be reconstructed by the time reversal, although this reconstruction is (expectably) somewhat less stable.

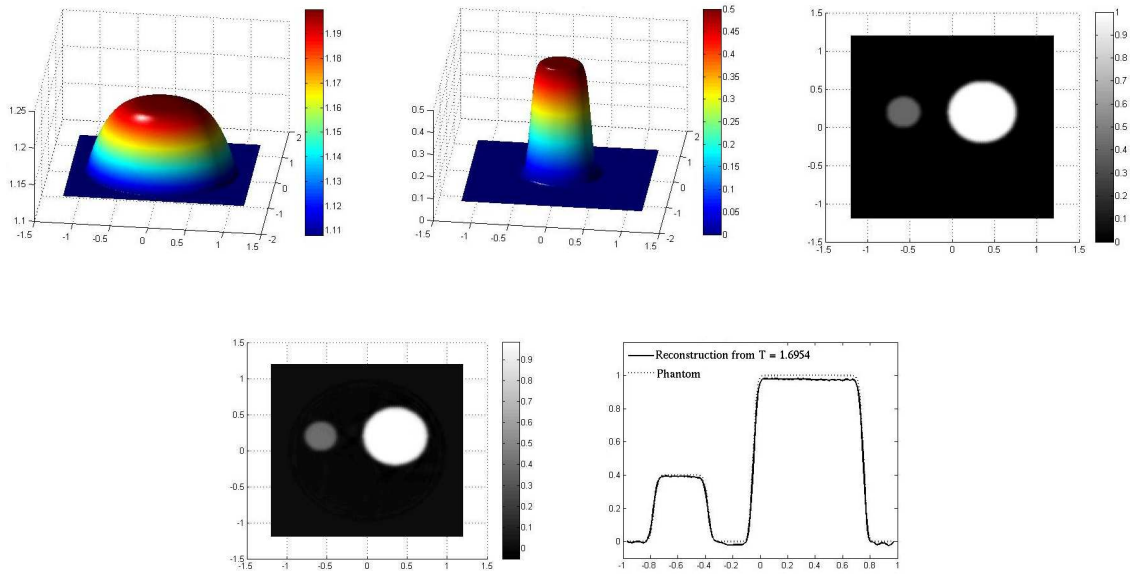

Fig. 2.6. Top: sound speed (left), initial velocity (middle), and density plot of the phantom (right). Bottom: reconstruction from time $T = 1.7$ (left) and a profile plot of the phantom and the reconstruction (right).

2.3.3. Trapping Variable Speed

We now turn to the worst case scenario: trapping speeds in $2D$. The simplest way to create a trapping speed is to use a speed with a "crater profile", see the graph on the left of Fig. 2.7. This radial sound speed equals $|x|$ in the annulus $\{x|0.5 < |x| < 1\}$ and is constant elsewhere. On Fig. 2.7 (right) is a phantom we used for the experiment with the "crater" speed. Reconstructions for $T = 4, 10,$ and 15 (it takes $t = 3.3863$ for the wave to go across the unit circle) are shown in Fig. 2.8. One clearly sees that although it takes a longer time, one still eventually reconstructs the phantom. There is, however, a phenomenon that can deteriorate the reconstructions

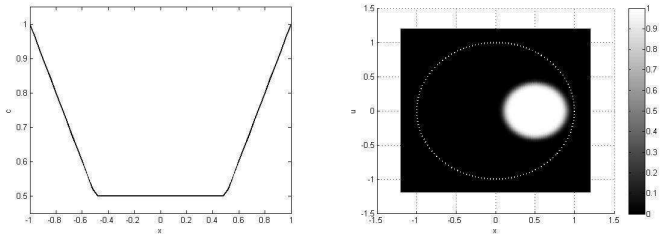

Fig. 2.7. A "crater" trapping speed (left) and the phantom used for reconstructions with this speed.

in the trapping situations, and which one should be aware of. We discuss it in the next sub-section.

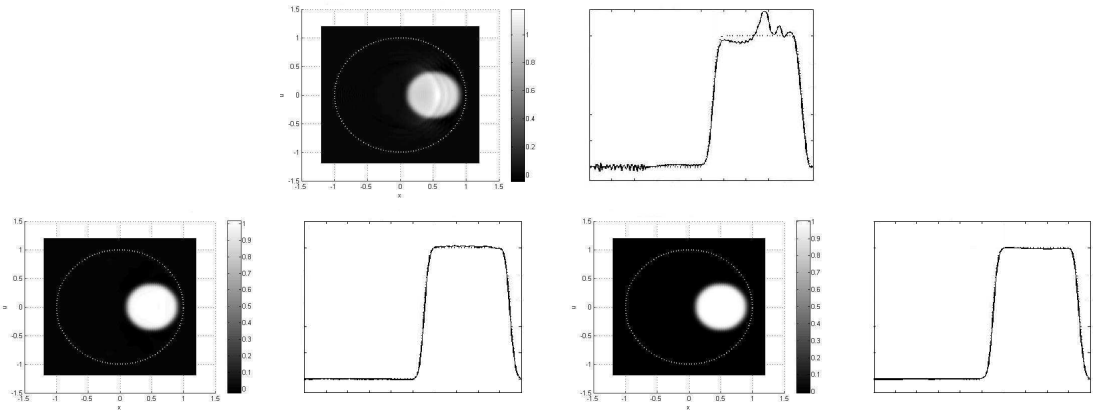

Fig. 2.8. The reconstructions for $T = 4, 10,$ and 15 with the "crater" trapping speed.

2.3.4. Limited View Effects Due to a Trapping Speed

Let us use again the same "trapping crater" speed (Fig. 2.7). On Fig. 2.9 a quarter-annulus phantom supported inside the trapping area and its reconstructions are shown. These reconstructions exhibit blurring of the radial sides of the phantom, a

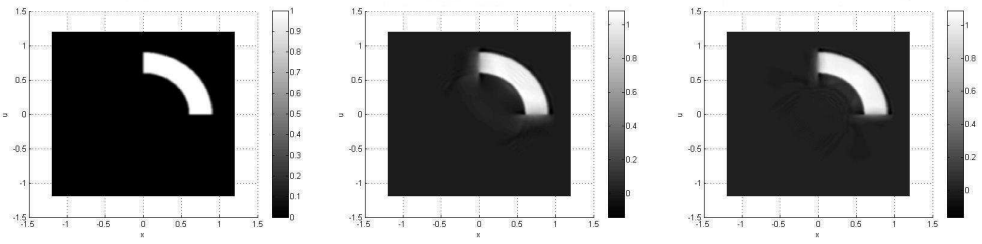

Fig. 2.9. A phantom and its reconstructions for $T = 4$ and 6 with the "crater" trapping speed.

phenomenon common to incomplete data problems [5], [6], [33], [42]–[45]. At a first glance, one may be led to think that the data is as complete as possible, since it is collected from the whole closed surface S. However, the "trapping crater" speed traps all rays corresponding to bicharacteristics that start at points (x_0, ξ_0) such that $x_0 \in \{x|0.5 < |x| < 1\}$ and $x_0 \perp \xi_0$. These rays are circular and thus never reach S. Hence, as we briefly discussed in Section 2.2, singularities corresponding to wave front vectors (x_0, ξ_0) will never reach the observation surface. This means, that the singularity of radial interfaces never show up as singularities in the measured data g. It has been argued then [5], [6], [33], [42]–[46] that such interfaces cannot be stably reconstructed and thus get blurred, no matter what reconstruction method is used. Since the rays that start being normal to other parts of the phantom's boundary

are not trapped, these parts are reconstructed stably. One should also note that if the phantom's boundaries inside the trapping ring were not radial, they would reconstruct sharply, since the related rays would reach S. This is confirmed by the reconstructions in Figs. 2.9, 2.10 and 2.11. It is interesting to notice what happens

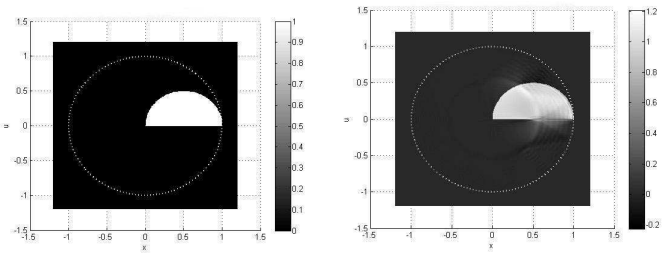

Fig. 2.10. A semi-circular phantom and its reconstructions for $T = 4$ with the "crater" trapping speed. The parts of the boundaries that are in the trapping ring, but such that the normal vectors produce non-trapped rays, are not blurred.

with the semi-circular phantom shown in Fig. 2.11. Here one notices a deterioration near the point on the semi-circular part, where the boundary is tangential to the radius. The reason is that the corresponding ray is trapped. In the reconstruction shown in Fig. 2.10, this effect did not appear, since the corresponding point was not in the trapping region and thus the relevant ray was escaping.

We now show another example of a trapping sound speed, which exhibits more severe trapping than the "crater"one. The "parabolic" speed equals $|x|^2 + 0.1$ inside the unit circle and is constant outside. In this case, for any x_0 there exists a cone of directions ξ_0, such that the ray corresponding to the bicharacteristic starting at x_0, ξ_0 is trapped [46]. On Fig. 2.12, the sound speed, a phantom, and its reconstruction are

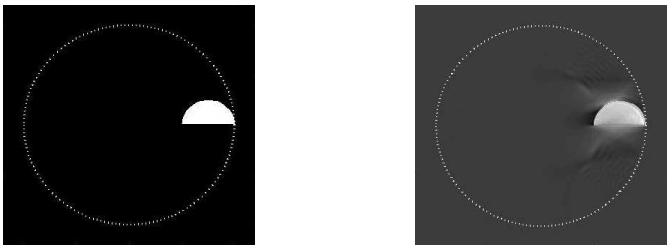

Fig. 2.11. A smaller semi-circular phantom and its reconstruction for $T = 4$ with the "crater" trapping speed. There is a noticeable blurring on the circular boundary at the point where the normal ray is trapped (i.e., where the circle is tangential to the radius).

shown. In this case, the blurring of the image is very strong, and the reconstructions do not improve with time, since the information about missing singularities never reaches the detectors.

2.3.5. Reconstruction with an Incorrect Speed

In the above discussions, we have always assumed that the sound speed was known, while it is often not. To overcome this, one either tries to recover somehow the speed (e.g., using transmission ultrasound tomography [47], [48]), or assumes some (usually constant) speed. As it has been noted in previous studies (e.g., [47]–[49]), it is easy to conclude that using an incorrect sound speed deteriorates both the amplitudes, as well as locations of features (e.g., of interfaces) of the image. The example shown below is provided to confirm this. On Fig. 2.13 a sound speed and a phantom are shown. The sound speed equals 1, except at four circular inclusions, where it takes

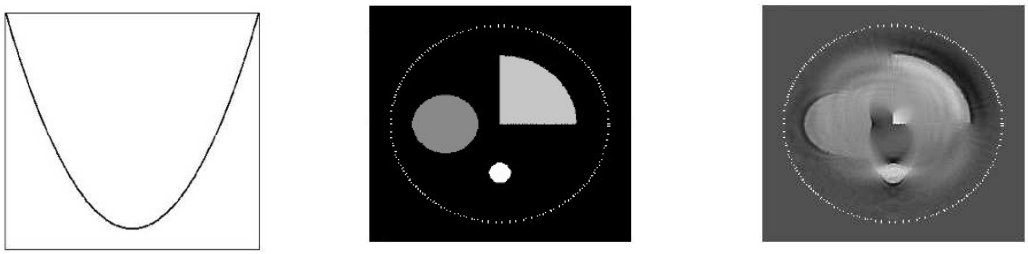

Fig. 2.12. Profile of the "parabolic" trapping speed (left), a phantom (middle) and its reconstruction from $T = 8$ (right).

values 1.5, 1.2, 0.8 and 0.7. Fig. 2.14 shows the reconstruction for $T = 2.5$ with the true speed and for $T = 5$ with the average speed $c = 1.0118$. One can easily see both types of deterioration mentioned above. This example also shows the viability of the time reversal in the case of more realistic sound speeds than used before (several different "tissues" are present).

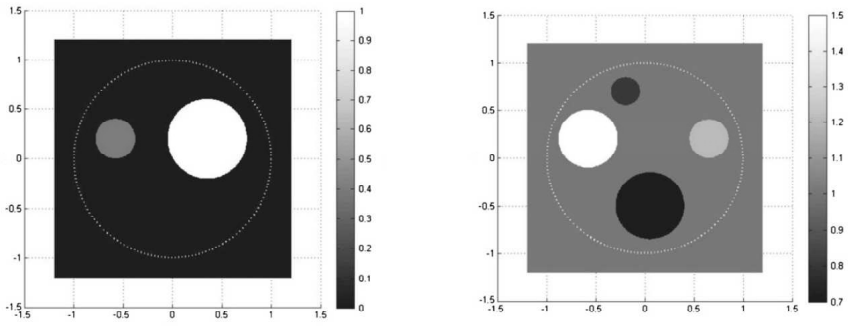

Fig. 2.13. A phantom (left) and a sound speed density plot (right).

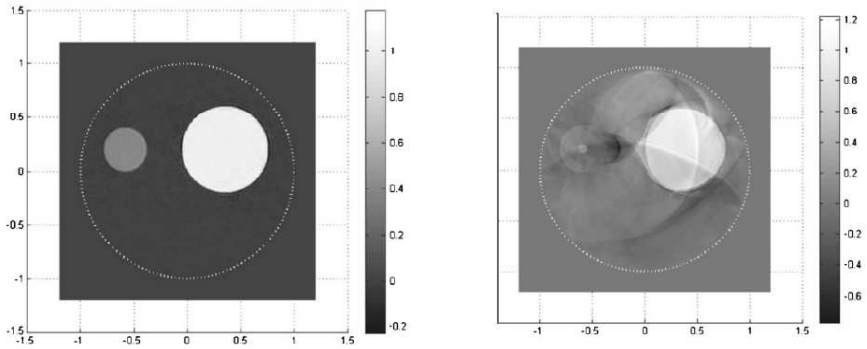

Fig. 2.14. The reconstruction with the correct speed map (left) and with the average speed (right). One observes both location shifts and amplitude deterioration of the image reconstructed with the average speed.

2.4. An Error Estimate

In this section we consider the error $e(x,t;T) = u(x,t) - v(x,t)$ that results from replacing the true solution of (2.13), $u(x,t)$, with the time reversal solution $v(x,t)$. Recall that $v(x,t)$ solves equation (2.14). Then, the error satisfies the following equation:

$$\begin{cases} e_{tt}(x,t;T) = c^2(x)\Delta e(x,t;T) & \text{in } B \times [0,T] \\ e(x,T;T) = p(x,T) \\ e_t(x,T;T) = p_t(x,T) \\ e|_S(x,t;T) = g_\varepsilon(x,t) \colon = (1-\varphi_\varepsilon(t))g(x,t) & \text{in } S \times [0,T]. \end{cases} \tag{2.15}$$

To make the role of ε more clear, we will specify our choice of a cut-off function $\varphi_\varepsilon(t)$. Let $\varphi(t)$ be a smooth function that equals to 1 on $(-\infty,-1]$ and vanishes on $[0,\infty)$. When $\varepsilon \leq 1$, we set $\varphi_\varepsilon(t) = \varphi((t-T)/\varepsilon)$ (Figure 2.15), and when $\varepsilon > 1$, we choose $\varphi_\varepsilon(t) = \varphi_1(t) = \varphi(t-T)$. This can also be written as $\varphi_\varepsilon(t) = \varphi((t-T)/\alpha)$, where $\alpha = \min\{\varepsilon, 1\}$.

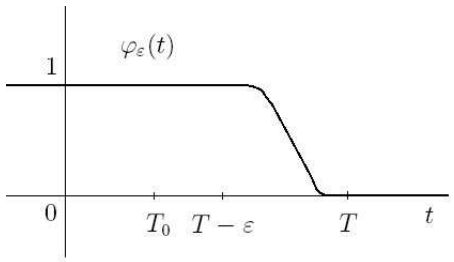

Fig. 2.15. A sketch of a cut-off function $\varphi_\varepsilon(t)$.

Let the sound speed $c(x)$ satisfy the following conditions: $c(x) > c > 0$, $c(x) \in C^\infty(\mathbb{R}^n)$, $c(x) - 1$ has a compact support and $c(x)$ is non-trapping. Suppose also

that the initial pressure $f(x)$ belongs to $L^2(\mathbb{R}^n)$ and is compactly supported (in B). Then the following theorem holds.

Theorem 3. *There exists T_0 such that for any $T > T_0$ and $\varepsilon > 0$ satisfying $T - \varepsilon > T_0$, the error $e(x, t; T)$ can be estimated as follows:*

- *for even dimensions n*

$$\max_{0\le t\le T}(\|e(t;T)\|_{H^1(B)} + \|e_t(t;T)\|_{L^2(B)}) \le C(\varepsilon)(T-\varepsilon)^{-n+1}\|f\|_{L^2(B)};$$

- *for odd dimensions n*

$$\max_{0\le t\le T}(\|e(t;T)\|_{H^1(B)} + \|e_t(t;T)\|_{L^2(B)}) \le C(\varepsilon)e^{-\delta(T-\varepsilon)}\|f\|_{L^2(B)}.$$

Here $C(\varepsilon) = C/\min\{\varepsilon^2, 1\}$, for some constant C depending on B and $c(x)$.

In particular,

$$\|e(0;T)\|_{H_0^1(B)} = \|f - v(0)\|_{H^1(B)} \le C(\varepsilon)(T-\varepsilon)^{-n+1}\|f\|_{L^2}, \text{ for even } n;$$

$$\|e(0;T)\|_{H_0^1(B)} = \|f - v(0)\|_{H^1(B)} \le C(\varepsilon)e^{-\delta(T-\varepsilon)}\|f\|_{L^2(B)}, \text{ for odd } n,$$

where $v(0, x)$ is the approximation to $f(x)$ obtained by time reversal with cut-off time T.

Proof. Let $E\colon H^s(S) \to H^{s+1/2}(B)$, $s > 0$, be the operator of harmonic extension that produces a harmonic function $E\phi$ in B from the Dirichlet boundary

data ϕ (e.g. [50]). Then $w\colon = e - Eg_\varepsilon$ satisfies

$$\begin{cases} w_{tt} - c^2(x)\Delta w = -Eg_{\varepsilon tt} & \text{in } B \times [0,T] \\ w(x,T) = p(x,T) - Eg_\varepsilon(x,T) = p(x,T) - Eg(x,T) \\ w_t(x,T) = p_t(x,T) - Eg_{\varepsilon t}(x,T) = p_t(x,T) - Eg_t(x,T) \\ w|_S(x,t) = 0 \quad \text{on } S \times [0,T], \end{cases} \tag{2.16}$$

As noted at the beginning of this section, there exists a time T_0 after which the solution $p(x,t)$ to (2.7) is infinitely smooth in $\bar{B} \times [T,\infty)$ for any $T > T_0$. Let us choose such T and let ε be such that $T - \varepsilon > T_0$. Then the right hand side and the initial conditions of (2.16) are infinitely smooth functions. Indeed, $p(x,T)$ is smooth, as we discussed above. Recall that g_ε is a smooth cut-off of the trace of $p(x,t)$. Moreover, it is supported in $[T-\varepsilon, T]$ – a time interval where $p(x,t)$ has already become infinitely smooth. Thus $Eg_\varepsilon(x,t)$ is a smooth function as well.

Next, we will apply the following theorem to the solution of (2.16).

Theorem 4. *(e.g.[19]) Assume that $G, H \in C^\infty(\bar{B}), F \in C^\infty(\bar{B} \times [0,T])$ and consider the initial boundary value problem*

$$\begin{cases} U_{tt} - c^2(x)\Delta U = F & \text{in } B \times (0,T] \\ U = G, \quad U_t = H & \text{on } B \times \{t = 0\}, \\ U = 0 & \text{on } S \times [0,T] \end{cases} \tag{2.17}$$

Here $c(x) \in C^\infty(\bar{B})$ is such that $c(x) \geq \theta$ for some $\theta > 0$. Let the following compatibility conditions hold for $l = 1, 2, ...$

$$
\begin{aligned}
&G_0 := G \in H_0^1(B),\ H_1 := H \in H_0^1(B),\\
&G_{2l} := \tfrac{d^{2l-2}F}{dt^{2l-2}}(\cdot,0) + c^2(x)\Delta G_{2l-2} \in H_0^1(B),\\
&H_{2l+1} := \tfrac{d^{2l-1}F}{dt^{2l-1}}(\cdot,0) + c^2(x)\Delta H_{2l-1} \in H_0^1(B).
\end{aligned}
\tag{2.18}
$$

Then (2.17) has a unique solution $U \in C^\infty(\bar{B}\times[0,T])$ *and*

$$
\max_{0\le t\le T}(\|U(t)\|_{H_0^1(B)} + \|U_t(t)\|_{L^2(B)}) \le
$$

$$
Ce^{C_1T}(\|F\|_{L^2(0,T;L^2(B))} + \|G\|_{H_0^1(B)} + \|H\|_{L^2(B)}),
$$

where the constants C *and* C_1 *depend on* B *and* $c(x)$.

We will verify that the conditions of Theorem 4 are satisfied by (2.16), noting that the compatibility conditions must hold at $t = T$, since the direction of time is reversed. Indeed, as we already discussed, $F = -Eg_{\varepsilon tt}$, $G = p(x,T) - Eg(x,T)$ and $H = p_t(x,T) - Eg_t(x,T)$ are infinitely smooth. It is clear that $G_0 = G$ and $H_1 = H$ belong to $H_0^1(\bar{B})$, as $g(x,t)$ has been defined to be the trace of $p(x,t)$. The higher order compatibility conditions hold as well:

$$
\begin{aligned}
G_{2l}(x) &= -\tfrac{d^{2l-2}}{dt^{2l-2}}Eg_{\varepsilon tt}(x,T) + c^2\Delta G_{2l-2} = -E\tfrac{d^{2l}}{dt^{2l}}g(x,T) + c^2\Delta G_{2(l-1)}\\
&= -E\tfrac{d^{2l}}{dt^{2l}}g(x,T) + (c^2\Delta)^l G_0 = -E\tfrac{d^{2l}}{dt^{2l}}g(x,T) + (c^2\Delta)^l p(x,T).
\end{aligned}
$$

We used here that $\Delta E = 0$.

As we already know that $p(x,T)$ is smooth in a neighborhood of T, we conclude that

$$
G_{2l}(x)|_S = \left[-\frac{d^{2l}}{dt^{2l}}p(x,T) + (c^2\Delta)^l p(x,T)\right]\Big|_S = 0,
$$

so $G_{2l} \in H_0^1(B)$ for $l = 1, 2...$ Similarly, one can check that $H_{2l+1} \in H_0^1(B)$ for $l = 1, 2....$

Before applying Theorem 4 to the solution of (2.16) we notice that the energy in B, defined by $\mathscr{E}(t) := \frac{1}{2}\int_B \left(|\nabla w|^2 + c^{-2}(x)\,|w_t|^2\right) dx$, stays constant in $[0, T-\alpha]$, where $\alpha = \min\{\varepsilon, 1\}$. Indeed, using (2.16) one easily shows that

$$\dot{\mathscr{E}}(t) = \int_B \left(\nabla w \cdot \nabla w_t + c^{-2} w_t w_{tt}\right)\, dx = -\int_B w_t c^{-2} E g_{\varepsilon tt}\, dx.$$

The last integral vanishes in $[0, T-\alpha]$, because $g_{\varepsilon tt} \equiv 0$ in $[0, T-\alpha]$ and therefore, $Eg_{\varepsilon tt} \equiv 0$ in this interval. Then, by Theorem (4), it follows that

$$\begin{aligned}
&\max_{0\le t\le T} \mathscr{E}(t) = \max_{T-\alpha\le t\le T} \mathscr{E}(t) \le C \max_{T-\alpha\le t\le T}\Big\{ \|w(t)\|^2_{H_0^1(B)} + \|w_t(t)\|^2_{L^2(B)} \Big\} \le \\
&C\Big\{ \|Eg_{\varepsilon tt}\|^2_{L^2(T-\alpha,T;L^2(B))} + \|p(\cdot,T) - \ Eg(\cdot,T)\|^2_{H_0^1(B)} \\
&\qquad\qquad + \|p_t(\cdot,T) - Eg_t(\cdot,T)\|^2_{L^2(B)} \Big\},
\end{aligned}$$

where C depends on $c(x)$ and B only, but not on T or ε. Here and in what follows C will denote various constants, all of them independent on T or ε.

As $w = e - Eg_\varepsilon$, the above inequalities imply

$$\begin{aligned}
&\max_{0\le t\le T}(\|e(t)\|_{H^1(B)} + \|e_t(t)\|_{L^2(B)}) \le \\
&C \max_{0\le t\le T}\left\{ \sqrt{\mathscr{E}(t)} + \|Eg_\varepsilon(t)\|_{H^1(B)} + \|Eg_{\varepsilon t}(t)\|_{L^2(B)} \right\} \le \\
&C\Big\{ \max_{T-\alpha\le t\le T}(\|Eg_\varepsilon(t)\|_{H^1(B)} + \|Eg_{\varepsilon t}(t)\|_{L^2(B)}) + \|Eg_{\varepsilon tt}\|_{L^2(T-\alpha,T;L^2(B))} \\
&\qquad + \|p(\cdot,T) - Eg(\cdot,T)\|_{H_0^1(B)} + \|p_t(\cdot,T) - Eg_t(\cdot,T)\|_{L^2(B)} \Big\}.
\end{aligned} \tag{2.19}$$

We used here that $Eg_\varepsilon \equiv 0$ in $[0, T-\alpha]$.

Let $\mathcal{T}\colon H^s(B) \to H^{s-1/2}(S)$, $s > 1/2$, denote the trace operator, which is known to be continuous (e.g.[19]). Then, using the continuity of the linear operators E and

$\mathcal{T}$, we can estimate the terms in (2.19) as follows.

$$\begin{aligned}\|Eg_\varepsilon(t)\|_{H^1(B)} \leq \;\; & C\|g_\varepsilon(t)\|_{H^{1/2}(S)} = C\|(1-\varphi_\varepsilon(t))\mathcal{T}p(t)\|_{H^{1/2}(S)} \leq \\ & C\|(1-\varphi_\varepsilon(t))p(t)\|_{H^1(B)} \leq C\|p(t)\|_{H^1(B)}.\end{aligned}$$

Similarly,

$$\begin{aligned}\|Eg_{\varepsilon t}(t)\|_{L^2(B)} \leq \;\; & \|Eg_{\varepsilon t}(t)\|_{H^1(B)} \leq C\|[(1-\varphi_\varepsilon)p(t)]_t\|_{H^1(B)} \leq \\ & \frac{C}{\alpha}\left\{\|p(t)\|_{H^1(B)} + \|p_t(t)\|_{H^1(B)}\right\},\end{aligned}$$

$$\begin{aligned}\|Eg_{\varepsilon tt}(t)\|_{L^2(B)} \leq \|Eg_{\varepsilon tt}(t)\|_{H^1(B)} \leq C\|[(1-\varphi_\varepsilon)p]_{tt}\|_{H^1(B)} \leq \\ \frac{C}{\alpha^2}\left\{\|p(t)\|_{H^1(B)} + \|p_t(t)\|_{H^1(B)} + \|p_{tt}(t)\|_{H^1(B)}\right\},\end{aligned}$$

$$\|Eg(T)\|_{H^1(B)} \leq C\|p(T)\|_{H^1(B)} \text{ and } \|Eg_t(T)\|_{L^2(B)} \leq C\|p_t(T)\|_{H^1(B)}.$$

Here we used that $\max_t |\varphi'_\varepsilon(t)| \leq C'/\alpha$ and $\max_t |\varphi''_\varepsilon(t)| \leq C''/\alpha^2$ for some constants C' and C''. Then, the error can be estimated by

$$\begin{aligned}&\max_{0\leq t\leq T}\left\{\|e(t)\|_{H^1(B)} + \|e_t(t)\|_{L^2(B)}\right\} \leq \\ &\frac{C}{\alpha^2}\left\{\max_{T-\alpha\leq t\leq T}\left(\|p(t)\|_{H^1(B)} + \|p_t(t)\|_{H^1(B)}\right) + \right. \\ &\qquad \|p\|_{L^2(T-\alpha,T;H^1(B))} + \|p_t\|_{L^2(T-\alpha,T;H^1(B))} + \|p_{tt}\|_{L^2(T-\alpha,T;H^1(B))} + \\ &\qquad\qquad \left.\|p(T)\|_{H^1(B)} + \|p_t(T)\|_{H^1(B)}\right\}\end{aligned} \tag{2.20}$$

Theorem 2 allows us to estimate the quantity inside the braces in right-hand side of above inequality by a factor of

$$\left\{ \max_{T-\alpha\le t\le T}[\eta_0(t)+\eta_1(t)] + \|\eta_0\|_{L^2(T-\alpha,T)} + \|\eta_1\|_{L^2(T-\alpha,T)} \right.$$
$$\left. + \|\eta_2\|_{L^2(T-\alpha,T)} + \eta_0(T) + \eta_1(T) \right\} \|f\|_{L^2(B)}$$

The functions $\eta_k(t)$ are monotonically decreasing, and $\eta_2(t) \le \eta_1(t) \le \eta_0(t)$ provided $t \ge 1$. Therefore, the right hand side of (2.20) is less than

$$C/\alpha^2 \left\{ \eta_0(T-\alpha) + \|\eta_0\|_{L^2(T-\alpha,T)} \right\} \|f\|_{L^2(B)}.$$

Taking into account that

$$\|\eta_0\|^2_{L^2(T-\alpha,T)} \le \alpha \max_{T-\varepsilon\le t\le T} \eta_0^2(t) \le \eta_0^2(T-\varepsilon),$$

we arrive to

$$\max_{0\le t\le T}(\|e(t)\|_{H^1(B)} + \|e_t(t)\|_{L^2(B)}) \le C(\varepsilon)\eta_0(T-\varepsilon)\|f\|_{L^2(B)},$$

where $\eta_0(T-\varepsilon) = (T-\varepsilon)^{-n+1}$ for even n, $\eta_0(T-\varepsilon) = e^{-\delta(T-\varepsilon)}$ for odd n, and $C(\varepsilon) = C/\min\{\varepsilon^2, 1\}$.

This proves the theorem.

2.5. Numerics of Errors

In this section we compare the errors of the numerical reconstructions of several phantoms to their estimates given in Theorem 3. The numerical simulations were done in two dimensions in media with variable sound speed. Both cases of non-trapping and trapping sound speed are considered.

In order to observe the behavior of the error of reconstruction with respect to the cut-off time T, multiple reconstructions from different cut-off times were made and the error was graphed on a logarithmic scale (see Appendix A). For the examples given in this section the spatial step-sizes varied case by case and were of the order of 10^{-2}. This corresponds to using several hundred detectors on the boundary of the unit circle.

On Figs. 2.16, 2.17, 2.18 and 2.19 we present various sound speeds and analyse the errors of reconstruction as functions of the cut-off time T. These figures show the particular sound speed, phantoms and the corresponding errors, but not actual reconstructions, as we already discussed those in length.

The plot of each error shows only values of the error after some minimal time, specific to each sound speed, after which it makes sense to apply the time reversal procedure.

Namely, this minimal time is taken to be the time a wave needs to cross the unit circle. After the error has decreased to a very small value it levels off, which is due to the discretization of the model. Thus, the error plots do not show these values. In order to easily observe the behavior of the error, the plots are made in a logarithmic scale and the corresponding linear regression interpolations are graphed. Figs. 2.16 and 2.17 show two examples of variable non-trapping sound speeds. In both cases two different phantoms are considered and the H^1 error of reconstruction as a function of the cut-off time T is plotted. In Theorem 3 we estimated that, in two dimensions, the error behaves as T^{-1}. The examples from Figs. 2.16 and 2.17 suggest even faster decay for those particular sound speeds. In the case shown in Fig. 2.16 the H^1 error decays as $T^{-1.3}$. The decay of the error in the second example (Fig. 2.17) is even faster: the plots suggest that it is of the order of $T^{-1.6}$. The next two examples deal with trapping sound speeds. In these cases the behavior of the error depends also on the initial condition, as there is no uniform local energy decay unless additional smoothness is assumed (e.g. [51], [52]). The L^2 errors shown in Figs. 2.18 and 2.19 decay as T increases.

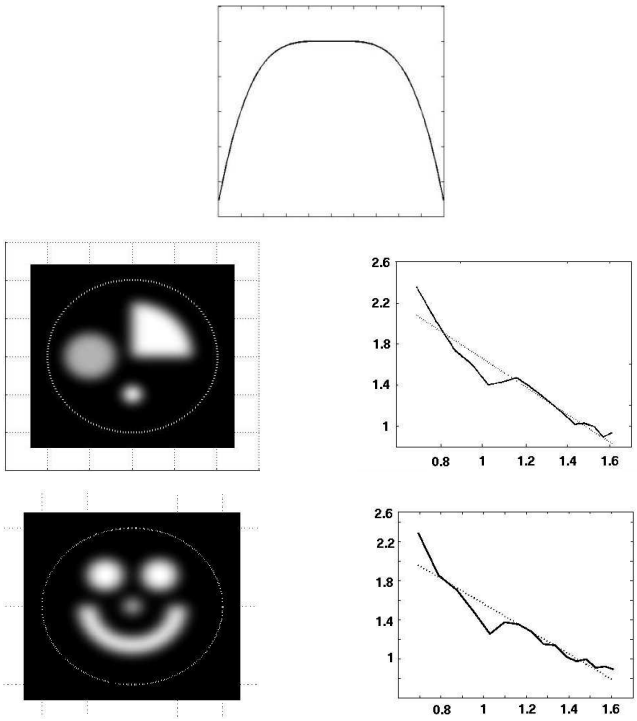

Fig. 2.16. Axial profile of a radial non-trapping sound speed (top), density plot of phantoms (left column) and the corresponding H^1 errors of reconstruction as functions of the cut-off time T (right column). The plots of the errors are in logarithmic scale, i.e. the horizontal axis represents $\ln T$ and the vertical axis represents $\ln(H^1$ error). The dotted lines show the linear regression interpolation of the error. The decay of the error in the first example (second row) is of the order of $T^{-1.36}$, in the second one (third row) it is $T^{-1.28}$.

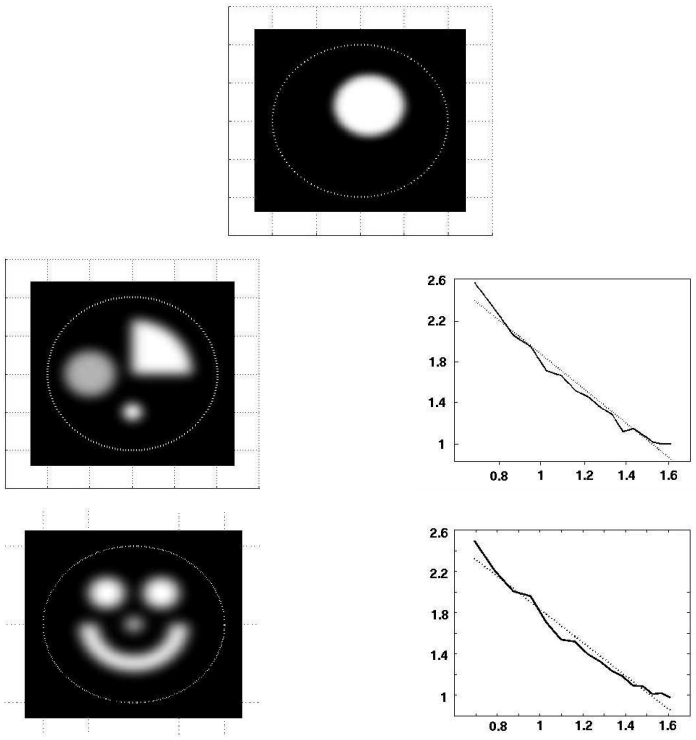

Fig. 2.17. Density plot of a non-radial non-trapping sound speed (top), density plot of phantoms (left column) and the corresponding H^1 errors of reconstruction as functions of the cut-off time T (right column). The plots of the errors are in logarithmic scale. The dotted lines show the linear regression interpolation of the error. The decay of the error in the first example (second row) is of the order of $T^{-1.68}$, in the second one (third row) it is $T^{-1.61}$.

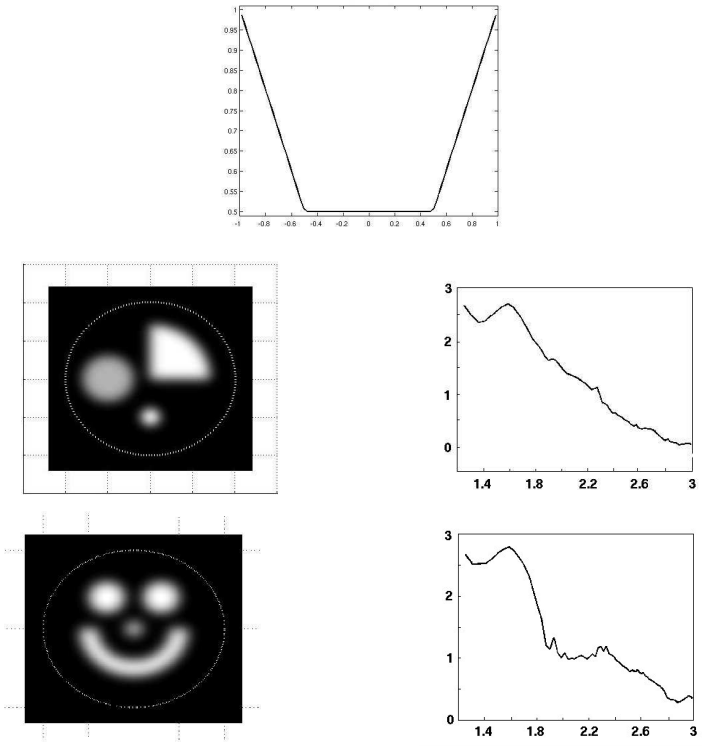

Fig. 2.18. Axial profile of a radial "trapping crater" sound speed (top), density plot of phantoms (left column) and the corresponding L^2 errors of reconstruction as functions of the cut-off time T (right column). The plots of the errors are in logarithmic scale.

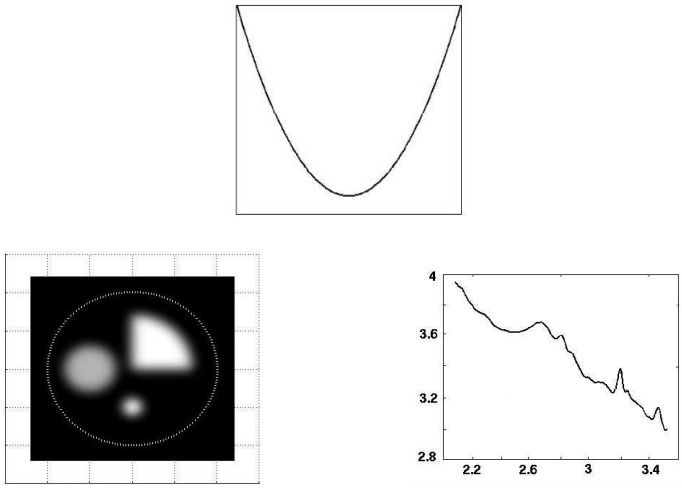

Fig. 2.19. Axial profile of a radial parabolic trapping sound speed (top), density plot of a phantom (left column) and the corresponding L^2 error of reconstruction as a function of the cut-off time T (right column). The plot of the error is in logarithmic scale.

2.6. Remarks

- We have concentrated on the errors resulting from the time reversal approximation. One is also interested in the stability of the method with respect to errors in the data. This question is answered by the standard results on stability of the mixed problem for the wave equation. If $n(x,t)$ is the error in the measured data $g(x,t)$, then a stability estimate should bound a norm of $\tilde{e}(x,0;T)$, where $\tilde{e}(x,t;T)$ solves the following mixed value problem:

$$\begin{cases} \tilde{e}_{tt}(x,t;T) = c^2(x)\Delta\tilde{e}(x,t;T) & \text{in } B\times[0,T] \\ \tilde{e}(x,T;T) = 0 \\ \tilde{e}_t(x,T;T) = 0 \\ \tilde{e}|_S(x,t;T) = n_\varepsilon(x,t) \colon = (1-\varphi_\varepsilon(t))n(x,t) & \text{in } S\times[0,T]. \end{cases} \tag{2.21}$$

Theorem 5. *Let $c(x)$ be smooth. Then the following stability estimate holds*

$$\|\tilde{e}(x,0;T)\|_{L^2(B)} \le C\|n(x,t)\|_{H^{1+\delta,1+\delta}([0,T]\times S)},$$

Here $H^{1+\delta,1+\delta}([0,T]\times S) = L^2(0,T;H^{1+\delta}(S)) \cap H^{1+\delta}(0,T;L^2(S))$ and $\delta > 0$.

Indeed, since $c(x)$ is assumed to be smooth, classical results (e.g. [53, Ch. 5, Sec. 5]) give the above inequality.

In a nutshell, the reconstruction is as stable as Radon inversion in the corresponding dimension.

- Our numerical examples show decay of the error also in the trapping speed case. One could probably get some estimates on this decay, assuming sufficient

smoothness of the function $f(x)$ to be reconstructed and using the results of [52]. It is unlikely that one can get such estimates without a smoothness assumption.

CHAPTER III

MATHEMATICAL PROBLEMS OF COMPTON CAMERA IMAGING

3.1. Introduction to Compton Camera Imaging

Compton cameras, also called Compton scatter cameras or electronically collimated cameras, are a type of highly efficient gamma cameras used in emission tomography. They have received a lot of attention since they were first proposed in [54] and a prototype was developed by Singh et al [55], [56]. Compton cameras were first suggested for use in nuclear medicine (SPECT), but they also have applications in astrophysics [57]–[59], monitoring nuclear power plants [60], [61], and other areas. Compton cameras offer several advantages over the conventional mechanically collimated (Anger) gamma cameras. The most significant of these is the dramatic increase of sensitivity – Compton cameras are reported to count at least an order of magnitude more photons than Anger cameras [62], [63]. Another advantage is the flexibility of geometrical design [60], [61], which even allows for hand-held devices [64], [65].

A compton camera consists of two detectors, which, in the usual physical setup, are planar and are placed one behind the other (see Fig. 3.1). A photon incident on the camera undergoes Compton scattering in the first detector, which records the location x_1 and the energy of interaction E_1. After the scattering, the photon is absorbed in the second detector, where the position x_2 and energy of absorption E_2 are measured. From the knowledge of E_1 and E_2 the scattering angle ψ can be

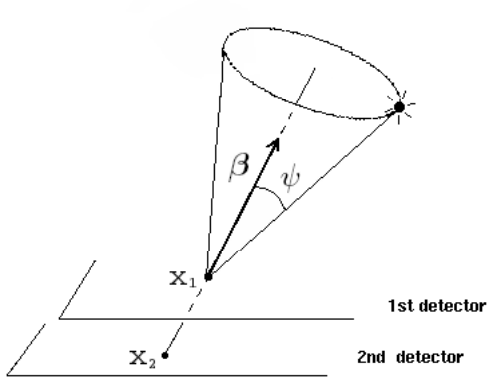

Fig. 3.1. Schematic representation of a Compton camera.

determined by the formula (e.g. [54], [66])

$$\cos\psi = 1 - \frac{mc^2 E_1}{(E_1 + E_2)E_2}. \tag{3.1}$$

Here m is the mass of an electron and c is the speed of light. The direction into which a photon scatters is given by $-\boldsymbol{\beta} := \dfrac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$. From the knowledge of $\boldsymbol{\beta}$ and the scattering angle ψ we conclude that the photon originated on the surface of the cone with central axis $\boldsymbol{\beta}$, vertex $\mathbf{x}_1$ and opening angle 2ψ. Therefore, although the exact incoming direction of the detected particle is not available, one knows a cone of such possible directions. The goal of Compton camera imaging is to recover the distribution of the radiation sources from this data.

The rest of this chapter is organized as follows. In Section 3.1.1 the mathematical problem of Compton camera imaging is formulated, and a brief discussion of known

reconstruction formulas in three dimensions id provided. In Section 3.2, the 2D problem is stated and several reconstruction formulas are presented. This section also contains numerical reconstructions based on the above methods. The last part of the chapter is devoted to the use of Compton cameras for detection of low emission point sources in the presence of a large background. This study was motivated by a homeland security project for detection of illicit radioactive materials.

The Matlab codes used in this chapter can be found in Appendix B.

3.1.1. The Cone Transform

Suppose that we want to image the distribution $f(\mathbf{y})$ of radioactivity sources inside a certain object in $\mathbb{R}^3$. From now on, we will assume that $f(\mathbf{y})$ is supported on one side of the Compton camera and that $\text{supp} f$ does not intersect the camera. If a large number of photons is detected, the Compton camera provides us with the projections, which we denote by $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$, i.e. the integrals of $f(\mathbf{y})$ on cones parameterized by a vertex $\mathbf{x}$ lying on the detector, central axis $\boldsymbol{\beta} \in S^3$ and half-angle $\psi \in [0, \pi]$ (see Fig.3.2) [66]–[68]:

$$\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi) = K(\psi) \int_0^{2\pi} \int_0^{\infty} f(\mathbf{x}+r\boldsymbol{\alpha}(\phi)) r \sin\psi \; dr \; d\phi, \text{ where } \boldsymbol{\alpha}\cdot\boldsymbol{\beta} = \cos\psi, \; \boldsymbol{\alpha} \in S^2. \tag{3.2}$$

Here $K(\psi)$ is the Klein-Nishina distribution of scattering angles. The function $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ provides the expectation of the number of hits, per unit time, from the particles moving along the cone. In particular, when the particle count is low, these values are not well determined by the data. This issue will be addressed in Section 3.4. An important observation is that the problem of inverting the cone

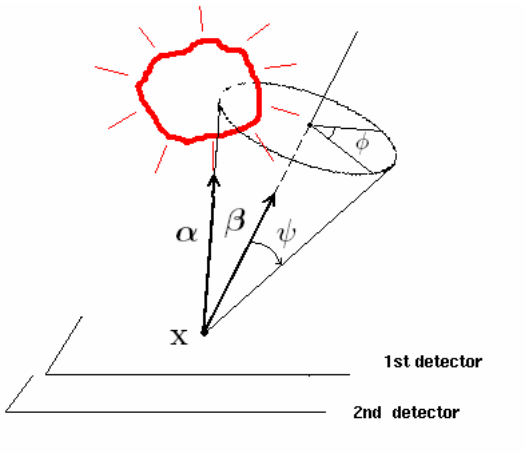

Fig. 3.2. A Compton cone with apex $\mathbf{x}$, central axis $\boldsymbol{\beta}$ and half-angle ψ.

transform $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ is overdetermined. Indeed, $f(\mathbf{y})$ is a function of three variables, while $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ depends on five parameters (here $\mathbf{x}$ is restricted to a 2D detector surface). Most authors consider restricted versions of the cone transform, reducing the number of parameters from five to three [66], [67], [69] . One little known paper [70] presents an elegant closed form inversion formula that uses four of the parameters. Only very recently, in [71], the whole set of data was used for reconstructions, albeit the transformation considered in this paper somewhat from the conical transform defined above.

3.1.2. Reconstruction Techniques in Compton Camera Imaging

Below we give a brief survey of reconstruction methods in Compton cameras imaging. Many of them reduce cone projections to Radon projections, i.e. integrals of f over planes.

3.1.2.1. Algebraic Reconstruction Techniques

Algebraic reconstruction techniques, as elsewhere in tomography, are widely used in Compton cameras imaging. This is partially due to the fact that analytic inversion formulas were not available until the second half of 1990's. Maximum likelihood methods were developed in [72]–[74]. A maximum likelihood method was also used in imaging with the COMPTEL telescope [57]. In [75] a matrix inversion technique was utilized to solve the problem for a specific detector geometry. Fast backprojection algorithms were developed in [76], [77].

3.1.2.2. Spherical Harmonics

A spherical harmonics solution was first proposed in [67]. For every point $\mathbf{x}$ on the first detector, and a fixed in advance half-angle ψ the authors relate the conical projection $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ to the Radon projection (integral) of f over the plane passing through x and perpendicular to β. A fast algorithm for computing the spherical harmonic series was also developed in this paper. Several other authors provided spherical harmonics solutions, however their model for Compton data differs from

(3.2), e.g. [78]–[80].

3.1.2.3. Closed Form Solutions

The first analytic inversion formula is due to Cree and Bones [66]. The authors considered only cones perpendicular to the detector plane, i.e. cones with central axis $\boldsymbol{\beta} = \mathbf{z}$. Thus, the Compton data depends only on the vertex $\mathbf{x} := (x, y)$ and the opening angle ψ. Let us denote $g(x, y, \psi) := \mathcal{C}\ f(\mathbf{x}, \mathbf{z}, \psi)$ and let $F(u, v, \cdot),\ G(u, v, \cdot)$ be the two-dimensional (x, y)-Fourier transforms of $f(x, y, z)$ and $g(x, y, \psi)$ respectively. Then the following relation holds:

$$F(u, v, \frac{\xi}{\sqrt{u^2 + v^2}}) = \mathcal{H}_0 \left[\frac{u^2 + v^2}{K(t)t\sqrt{1 + t^2}} G(u, v, t) \right] \Bigg|_{t=\xi}, \tag{3.3}$$

where $\xi = z\sqrt{u^2 + v^2}$ and $t = \tan\psi$. Here $\mathcal{H}_0$ is the zero-order Hankel transform

$$\mathcal{H}_0 g(\rho) = 2\pi \int_0^\infty g(r) r J_0(2\pi r\rho)\ dr,$$

and $J_0(\gamma) = (2\pi)^{-1} \int_0^{2\pi} e^{i\gamma\cos\phi}\ d\phi$ is the zero-order Bessel function. Later, in [69], more properties of the same restricted cone transform were given.

In [71] the authors use yet another mathematical formulation for the forward problem, which accounts for the efficiency of the detector at different incident angles of particles. They extended the approach of Cree and Bones to show that the radioactivity distribution can be reconstructed from any family of cones that have central axes β such that the angle α between β and the $\mathbf{z}$-axis is fixed. Then averaging of the solutions for different values of α is employed, thus making use of all

available data and reducing noise.

In [70] the following relation between the cone and Radon transforms of f is given:

$$HRf(\boldsymbol{\beta}, \mathbf{x}\cdot\boldsymbol{\beta}) := \frac{1}{\pi}\int_{-\infty}^{\infty} Rf(\boldsymbol{\beta}, t)h(\mathbf{x}\cdot\boldsymbol{\beta}-t)\, dt = -\frac{1}{\pi}\int_{0}^{\pi} g(\mathbf{x}, \boldsymbol{\beta}, \psi)h(\cos\psi)\, d\psi \quad (3.4)$$

Here H denotes the Hilbert transform and R is the Radon transform in three dimensions (see Section 1.1). Note that four out of five variables are used and that the $HRf(\boldsymbol{\beta}, \mathbf{x}\cdot\boldsymbol{\beta})$ is constant on the line $L_c := \{\mathbf{x} \in \text{detector plane} | \mathbf{x}\cdot\boldsymbol{\beta} = c\}$, for any fixed constant c. The last observation shows that it is possible to reconstruct f if the vertices of cones $\mathbf{x}$ are restricted to a curve. This also allows for averaging of the solution on the lines L_c. In Section 3.2.2 we will consider the two-dimensional analog of this formula.

3.2. Compton Cameras in Two Dimensions

In the two dimensional setting we assume that the detectors $\mathbf{x}$ lie on a line, which we call the *detector line*. A cone in two dimensions is defined by a point $\mathbf{x}$, a cental axis $\boldsymbol{\beta}$ and a half-angle ψ. Such cones simply consist of two half-lines intersecting at a point. More precisely, let $\boldsymbol{\beta} = (\cos\beta, \sin\beta) \in S^1, \beta \in [0, \pi]$ be the central axis of a cone with opening half-angle $\psi \in [0, \pi]$. The two half-lines that constitute the cone, pass through the detector point $\mathbf{x}$ and are given by $\{\mathbf{y} \in \mathbb{R}^2 : \mathbf{y} = r\boldsymbol{\alpha}_i, r \geq 0\}$, $i = 1, 2$, where $\boldsymbol{\alpha}_1 = (\cos(\beta - \psi), \sin(\beta - \psi))$ and $\boldsymbol{\alpha}_2 = (\cos(\beta + \psi), \sin(\beta + \psi))$, see Fig. 3.3. In what follows we will assume uniform distribution of scattering angles, that is $K(\psi) \equiv 1$. Then the two dimensional cone transform $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ is the

projection of $f(\mathbf{y})$ onto these lines:

$$\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi) = \int_0^\infty \left(f(\mathbf{x} + r\boldsymbol{\alpha}_1) + f(\mathbf{x} + r\boldsymbol{\alpha}_2)\right)\, dr = \mathcal{D}f(\mathbf{x}, \boldsymbol{\alpha}_1) + \mathcal{D}f(\mathbf{x}, \boldsymbol{\alpha}_2). \quad (3.5)$$

Here $\mathcal{D}f(\mathbf{x}, \alpha) = \int_0^\infty f(\mathbf{x} + r\boldsymbol{\alpha})\, dr$ denotes the divergent beam (or fanbeam) transform of f.

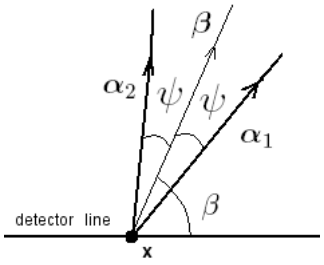

Fig. 3.3. Two-dimensional cone with apex $\mathbf{x}$, central axis $\boldsymbol{\beta}$ and half-angle ψ.

For convenience, we will sometimes express $\mathcal{C}f$ and $\mathcal{D}f$ as functions of the angle β instead of the unit vector $\boldsymbol{\beta}$. Thus, we can write (3.5) as

$$\mathcal{C}f(\mathbf{x}, \beta, \psi) = \mathcal{D}f(x, \beta - \psi) + \mathcal{D}f(x, \beta + \psi). \quad (3.6)$$

The problem of inverting the two-dimensional transform is overdetermined as well, because $\mathcal{C}f(\mathbf{x}, \boldsymbol{\beta}, \psi)$ depends on three parameters and $f(\mathbf{y})$ only on two.

3.2.1. Reconstruction Formula by Basko et al.

The author is aware of only few papers devoted to compton cameras in two dimensions. In [81], [82] subsets of the cone projections of $f(x)$ are represented as line integrals of the function f added with its mirrored shear transformation. In order to explain this approach in more detail, let us introduce the variables $k_1 = \cot(\beta + \psi), k_2 = \cot(\beta - \psi)$. The cone transform can be written as

$$\mathcal{C}f(x, \beta, \psi) = \int_0^\infty f(x + k_1 z, z)\, dz + \int_0^\infty f(x + k_2 z, z)\, dz, \tag{3.7}$$

where the $x-$ axis coinsides with the detector line and the $z-$ axis is perpendicular to the detectors. Let us fix the sum $k_1 + k_2 = K$ and define the function

$$f_K(x, z) = \begin{cases} f(x, z), & \text{if } z \geq 0 \\ f(x - Kz, -z), & \text{if } z < 0. \end{cases} \tag{3.8}$$

Then, $\mathcal{C}f(x, \beta, \psi) = \int_{-\infty}^{\infty} f_K(x + k_1 z, z)$. In this way, for each fixed value of K, the cone projections are represented as line integrals of a certain function, so all that remains is to invert the Radon transform. The reconstructed image will contain the function f above the detector line and a mirrored shear transformation of f below the detectors. Obviously, by fixing the parameter K, not all available data is used. The case when $K = 0$ corresponds to using only those cones with central axes orthogonal to the detector line.

Although there are not many papers devoted to the two-dimensional Compton problem, most inversion formulas for the three dimensional case should have analogues in two dimensions.

3.2.2. Three Inversion Formulas

3.2.2.1. Method A

The following relation makes use of all parameters and is an analog of the three dimensional result presented in [70].

Theorem 6. *Let the closed unit ball lie on one side of and away from the detector line, and let* $f(x) \in H_0^{1/2}(B^2)$. *Denote* $h(t) = 1/t$. *Then*

$$H\mathcal{R}f(\mathbf{x}\cdot\boldsymbol{\beta},\boldsymbol{\beta}) = \frac{1}{\pi}\int_{-\infty}^{\infty}\mathcal{R}f(\boldsymbol{\beta},t)h(\mathbf{x}\cdot\boldsymbol{\beta}-t)\,dt = -\frac{1}{\pi}\int_0^{\pi}\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},\psi)h(\cos\psi)\,d\psi \tag{3.9}$$

Here H *is the Hilbert transform in the first (scalar) variable and the integrals are understood in the principal value sense (see Section 1.1).*

Proof. We will consider the following regularizations of the above improper integrals.

$$\int_{-\infty}^{\infty}\mathcal{R}f(t,\boldsymbol{\beta})h(\mathbf{x}\cdot\boldsymbol{\beta}-t)\,dt = \int_0^{\infty}\frac{\mathcal{R}f(\mathbf{x}\cdot\boldsymbol{\beta}-t,\boldsymbol{\beta})-\mathcal{R}f(\mathbf{x}\cdot\boldsymbol{\beta}+t,\boldsymbol{\beta})}{t}\,dt \tag{3.10}$$

$$\int_0^{\pi}\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},\psi)h(\cos\psi)\,d\psi = \int_{\pi/2}^{\pi}\frac{\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},\psi)-\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},-\psi+\pi)}{\cos\psi}\,d\psi \tag{3.11}$$

Thus, we need to show that

$$\begin{aligned}\int_{\pi/2}^{\pi}\frac{\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},\psi)-\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},-\psi+\pi)}{\cos\psi}\,d\psi = \\ -\int_0^{\infty}\frac{\mathcal{R}f(\mathbf{x}\cdot\boldsymbol{\beta}-t,\boldsymbol{\beta})-\mathcal{R}f(\mathbf{x}\cdot\boldsymbol{\beta}+t,\boldsymbol{\beta})}{t}\,dt\end{aligned} \tag{3.12}$$

We first address the question of whether the integrals in (3.10) and (3.11) are well defined. From the property (1.9) of Radon transforms it follows that $\mathcal{R}f(s,\boldsymbol{\beta})$ be-

longs to the space $H^1(\mathbb{R} \times S^1)$. It is a standard result (e.g. [19, Chapter 5.6.2]) that $H^1(\mathbb{R})$ is contained in the space of Hölder continuous functions with exponent $1/2$. Therefore, for each fixed $\boldsymbol{\beta}$, $\mathcal{R}f(s, \boldsymbol{\beta})$ is Hölder continuous, so the integral in (3.10) is well defined (see Section 1.1).

The integral in (3.11) is the canonical regularization of a singular integral, in the sense defined in [83]. Now we will show why (3.11) is well defined. The relations between fanbeam coordinates $(\mathbf{x}, \boldsymbol{\alpha})$ and parallel beam coordinates $(s, \boldsymbol{\omega})$, where $\boldsymbol{\omega} := (\cos\phi, \sin\phi)$ are

$$\phi = \alpha - \pi/2, \quad s = \mathbf{x} \cdot \boldsymbol{\omega}$$

Using the definition of cone transform, (3.6), and the assumption that f is supported on one side of the detector, we can write

$$\mathcal{C}f(\mathbf{x}, \beta, \psi) = \mathcal{R}f(\mathbf{x} \cdot \boldsymbol{\omega}, \beta - \psi - \pi/2) + \mathcal{R}f(\mathbf{x} \cdot \boldsymbol{\omega}, \beta + \psi - \pi/2). \tag{3.13}$$

From (1.9) it follows that, $\mathcal{C}f(\mathbf{x}, \beta, \psi)$ is in $H^1(\mathbb{R})$ with respect to ψ, so it is Hölder continuous in ψ.

Let us change the variables in the integral on the right hand side of 3.9: $\tilde{\psi} = \psi - \pi/2$. Then the integral reads as follows

$$-\int_{-\pi/2}^{\pi/2} \frac{\mathcal{C}f(\mathbf{x}, \beta, \psi - \pi/2)}{\sin\psi}\, d\psi, \tag{3.14}$$

where we have written ψ instead of $\tilde{\psi}$.

We note that the function $p(x) := \dfrac{\psi}{\sin\psi}$ is infinitely differentiable in $(0, \pi/2)$ and we can write $\dfrac{1}{\sin\psi} = p(\psi)\dfrac{1}{\psi}$.

The integral (3.14) can be regularized using (1.5)

$$-\int_{-\pi/2}^{\pi/2} \frac{p(\psi)\mathcal{C}f(\mathbf{x},\beta,\psi-\pi/2)}{\psi}\,d\psi = \\ -\int_{0}^{\pi/2} \frac{p(\psi)\,\mathcal{C}f(\mathbf{x},\beta,\psi+\pi/2)-p(-\psi)\,\mathcal{C}f(\mathbf{x},\beta,-\psi+\pi/2)}{\psi}\,d\psi. \tag{3.15}$$

This integral is well defined, because $\mathcal{C}f(\mathbf{x},\beta,\psi)$ is Hölder continuous with respect to ψ, and therefore so is $p(\psi)\,\mathcal{C}f(\mathbf{x},\beta,\psi)$. After changing variables $\tilde{\psi}=\psi+\pi/2$ we come to equation (3.11)

Now, we write (3.12) as follows

$$\int_{\pi/2}^{\pi} \frac{\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},\psi)-\mathcal{C}f(\mathbf{x},\boldsymbol{\beta},-\psi+\pi)}{\cos\psi}\,d\psi = \\ \int_{\pi/2}^{\pi} \frac{\mathcal{D}f(\mathbf{x},\boldsymbol{\beta}+\psi)-\mathcal{D}f(\mathbf{x},\boldsymbol{\beta}-\psi+\pi)}{\cos\psi}\,d\psi + \\ \int_{\pi/2}^{\pi} \frac{\mathcal{D}f(\mathbf{x},\boldsymbol{\beta}-\psi)-\mathcal{D}f(\mathbf{x},\boldsymbol{\beta}+\psi-\pi)}{\cos\psi}\,d\psi = \\ \int_{\pi/2}^{\pi}\int_{0}^{\infty} \frac{1}{r\cos\psi}\,[f(\mathbf{x}+r(\cos(\beta+\psi),\sin(\beta+\psi)))- \\ f(\mathbf{x}+r(\cos(\beta-\psi+\pi),\sin(\beta-\psi+\pi)))]\;r dr d\psi + \\ \int_{\pi/2}^{\pi}\int_{0}^{\infty} \frac{1}{r\cos\psi}\,[f(\mathbf{x}+r(\cos(\beta-\psi),\sin(\beta-\psi)))- \\ f(\mathbf{x}+r(\cos(\beta+\psi-\pi),\sin(\beta+\psi-\pi)))]\;r dr d\psi. \tag{3.16}$$

If we consider the (positively oriented) coordinate system $(\boldsymbol{\beta},\boldsymbol{\beta}^{\perp})$, the last two integrals can be written as integrals on the second (II) and third (III) quadrant of the plane. Indeed, if we make the change of variables $\mathbf{y}_1 = r(\cos(\beta+\psi),\sin(\beta+\psi))$

and $\mathbf{y}_2 = r(\cos(\beta-\psi), \sin(\beta-\psi))$ in the first and second integral respectively (see Fig.3.4), the expression (3.16) equals

$$\int_{II} \frac{1}{\mathbf{y}_1 \cdot \boldsymbol{\beta}} (f(\mathbf{x}+\mathbf{y}_1) - f(\mathbf{x}-\bar{\mathbf{y}}_1))\, d\mathbf{y}_1 + \int_{III} \frac{1}{\mathbf{y}_2 \cdot \boldsymbol{\beta}} (f(\mathbf{x}+\mathbf{y}_2) - f(\mathbf{x}-\bar{\mathbf{y}}_2))\, d\mathbf{y}_2. \quad (3.17)$$

Here $\bar{\mathbf{y}}$ denotes the reflection of $\mathbf{y}$ with respect to $\boldsymbol{\beta}$, i.e. if $\mathbf{y}$ is given by $\mathbf{y} = t\boldsymbol{\beta} + s\boldsymbol{\beta}^{\perp}$, then $\bar{\mathbf{y}} = t\boldsymbol{\beta} - s\boldsymbol{\beta}^{\perp}$.

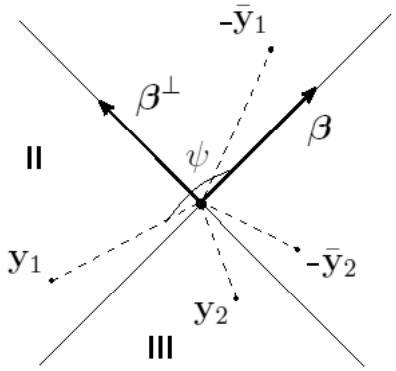

Fig. 3.4. Variables of intergation: $\mathbf{y}_1$ belongs to the second quadrant and $\mathbf{y}_2$ - to the third.

The integrals in (3.17) can be written as

$$\int\limits_{\mathbf{y}\cdot\boldsymbol{\beta}<0} \frac{1}{\mathbf{y}\cdot\boldsymbol{\beta}}(f(\mathbf{x}+\mathbf{y})-f(\mathbf{x}-\bar{\mathbf{y}}))\,d\mathbf{y} =$$

$$\int_{-\infty}^{0}\int_{-\infty}^{\infty}\frac{1}{t}\left[f(\mathbf{x}+t\boldsymbol{\beta}+s\boldsymbol{\beta}^{\perp})-f(\mathbf{x}-t\boldsymbol{\beta}+s\boldsymbol{\beta}^{\perp})\right]\,dsdt =$$

$$-\int_{0}^{\infty}\frac{1}{t}\left\{\int_{-\infty}^{\infty}f((\boldsymbol{\beta}\cdot\mathbf{x}-t)\boldsymbol{\beta}+s\boldsymbol{\beta}^{\perp})\,ds-\int_{-\infty}^{\infty}f((\boldsymbol{\beta}\cdot\mathbf{x}+t)\boldsymbol{\beta}+s\boldsymbol{\beta}^{\perp})\,ds\right\}\,dt =$$

$$-\int_{0}^{\infty}\frac{\mathcal{R}f(\boldsymbol{\beta}\cdot\mathbf{x}-t,\boldsymbol{\beta})-\mathcal{R}f(\boldsymbol{\beta}\cdot\mathbf{x}+t,\boldsymbol{\beta})}{t}\,dt. \tag{3.18}$$

In the above string of equations we used the representation of $\mathbf{x}$ in the coordinate system $(\boldsymbol{\beta},\boldsymbol{\beta}^{\perp})$ and made the change of variables $\boldsymbol{\beta}^{\perp}\cdot\mathbf{x}+s\rightarrow s$. □

Corollary 7. *Theorem 6 provides an inversion formula for the cone transform.*

Indeed, computing the integral of the right hand side of (3.9) one recovers $H\mathcal{R}f$, where $\mathcal{R}$ is the 2D Radon transform, and H is the Hilbert transform with respect to the linear variable. Then, the filtered backprojection formula (1.15) implies that differentiating the right hand side of (3.9) with respect to the linear variable and backprojecting, one recovers the function f.

Remark 8. *On the left hand side of (3.9) one sees $\mathcal{R}f(s,\boldsymbol{\beta})$, where $s=\mathbf{x}\cdot\boldsymbol{\beta}$. Thus, if one infinite detector line is used, if $\boldsymbol{\beta}$ is perpendicular to the detectors, we would only know $\mathcal{R}f(s,\boldsymbol{\beta})$ for $s=0$. If the detector line is finite, as is the case in all implementations, we would miss an even bigger chunk of data. One possible way to obtain the complete Radon data in $(s,\boldsymbol{\beta})\in[-1,1]\times S^{1}$ is, for example, to use three finite size detectors placed on sides of the square, containing the object.*

3.2.2.2. Method B

Another, simpler way of obtaining Radon data from cone data is presented below. Let us denote by $u(\mathbf{x}, \alpha)$ the integral of f along the half-line starting at the point $\mathbf{x}$ in direction of the unit vector $\boldsymbol{\alpha}$ (i.e. the fanbeam projection of f):

$$u(\mathbf{x}, \boldsymbol{\alpha}) = \mathcal{D}f(\mathbf{x}, \boldsymbol{\alpha}) := \int_0^\infty f(\mathbf{x} + r\boldsymbol{\alpha})\, dr. \tag{3.19}$$

Let us fix a detector position $\mathbf{x}$ and a direction $\boldsymbol{\alpha}$ and try to recover the fanbeam data $u(\mathbf{x}, \boldsymbol{\alpha}_0)$. For any $\psi \in [-\pi, \pi]$ we can write $\mathcal{C}f(\mathbf{x}, \beta, |\psi|) = u(\mathbf{x}, \beta - |\psi|) + u(\mathbf{x}, \beta + |\psi|)$. Thus, the following relation holds

$$u(\mathbf{x}, \alpha_0) = \mathcal{C}f(\mathbf{x}, \alpha_0 + \psi, |\psi|) - u(\alpha_0 + 2\psi)$$

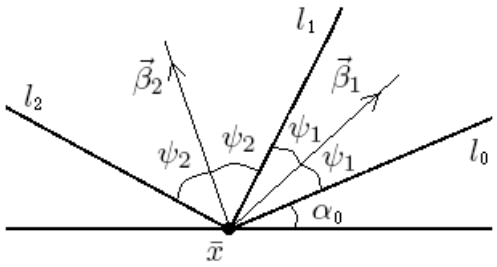

Fig. 3.5. In order to determine the integral of f along the line l_0, add the integral on cone consisting of l_0 and l_1 to the integral on the cone determined by l_0 and l_2. Then subtract the integral on the cone determined by l_1 and l_2.

It is easily seen from Fig. 3.5 that for any two half-angles $\psi_1, \psi_2 \in [-\pi, \pi]$ the

fanbeam data $u(\mathbf{x}, \alpha_0)$ can be obtained from cone data as follows:

$$u(\mathbf{x}, \alpha_0) = \frac{1}{2}\left[\mathcal{C}f(\mathbf{x}, \alpha_0 + \psi_1, |\psi_1|) + \mathcal{C}f(\mathbf{x}, \alpha_0 + \psi_2, |\psi_2|) - \mathcal{C}f(\mathbf{x}, \alpha_0 + \psi_1 + \psi_2, |\psi_1 - \psi_2|)\right] \tag{3.20}$$

The angles ψ_1 and ψ_2 were arbitrarily chosen, so one could average on ψ_1 and ψ_2 in order to use all available data. This averaging also helps reducing the effects of large random background or noise.

If averaging on half-angles ψ_1 and ψ_2 is used, the method described above is computationally expensive. We propose a third reconstruction method, which is fast and makes use of all available data.

3.2.2.3. Method C

Recall that the radioactivity function $f(\mathbf{y})$ is compactly supported and supp$f$ lies on one side and away from the detector line. Let us fix a detector $\mathbf{a}$. Then, $u(\mathbf{a}, \alpha)$ is supported in $(0, \pi)$. The function $u(\mathbf{a}, \alpha)$ can be extended periodically in α from $[0, 2\pi]$ to $(-\infty, \infty)$. Let us define

$$\tilde{u}(\mathbf{a}, \alpha) = \pi^{-1} \int_{-\alpha}^{\pi-\alpha} \mathcal{C}f(\mathbf{a}, \alpha + \psi, |\psi|)\, d\psi \tag{3.21}$$

We can write,

$$\tilde{u}(\mathbf{a},\alpha) = \pi^{-1}\{\int_{-\alpha}^{\pi-\alpha} u(\mathbf{a},\alpha)\, d\psi + \int_{-\alpha}^{\pi-\alpha} u(\mathbf{a},\alpha+2\psi)\, d\psi\} = u(\mathbf{a},\alpha) + (2\pi)^{-1}\int_0^{2\pi} u(\mathbf{a},\alpha)\, d\alpha \tag{3.22}$$

In the above equation, the periodicity of $u(\mathbf{a},\alpha)$ in α was used. Then,

$$\tilde{u}(\mathbf{a},\alpha) = u(\mathbf{a},\alpha) + (4\pi)^{-1}\int_{|\boldsymbol{\omega}|=1}\int_{-\infty}^{\infty} f(\mathbf{a}+r\boldsymbol{\omega})\, dr d\boldsymbol{\omega} = \mathcal{D}f(\mathbf{a},\alpha) + (4\pi)^{-1}\mathcal{R}^{\sharp}\mathcal{R}f(\mathbf{a}) \tag{3.23}$$

Recall that $\mathcal{R}^{\#}$ denotes the backprojection operator, $\mathcal{R}^{\#}g(\mathbf{y}) = \int_{|\boldsymbol{\omega}|=1} g(\mathbf{y}\cdot\boldsymbol{\omega},\boldsymbol{\omega})\, d\boldsymbol{\omega}$. Let us consider the second term on the right-hand side of (3.23) can be written as $\mathcal{D}[(4\pi)^{-1}\mathcal{R}^{\sharp}\mathcal{R}f(\mathbf{a})\delta(\mathbf{a}-\mathbf{y})]$. Therefore, when the inverse fanbeam transform is applied to $\tilde{u}$, the result equals the function $f(\mathbf{y})$ we are after, plus a weighted $\delta-$ function supported at the detector lines. Hence, we recover correctly functions f supported away from the detectors.

3.3. Numerical Experiments

In this section we present reconstructions of 2D numerical phantoms via the methods presented in Section 3.2.2.

3.3.1. Method A

We first note that the use of one finite size detector array results in limited view data. This means that not all data needed for correct reconstruction of the phantom was

available. Corollary 7 describes how to reduce cone data to Radon data. However, the Radon data we obtain is not complete, i.e. we know only the integrals of the phantom f along lines intersecting the detector. It is known in limited data X-ray tomography that the values of the phantom will not be reconstructed correctly. However, certain singularities of the phantom will be stably recovered. Namely, the reconstructed image will contain only those singularities which have tangent line crossing the detector array [43], [44]. The rest of the singularities will be smeared.

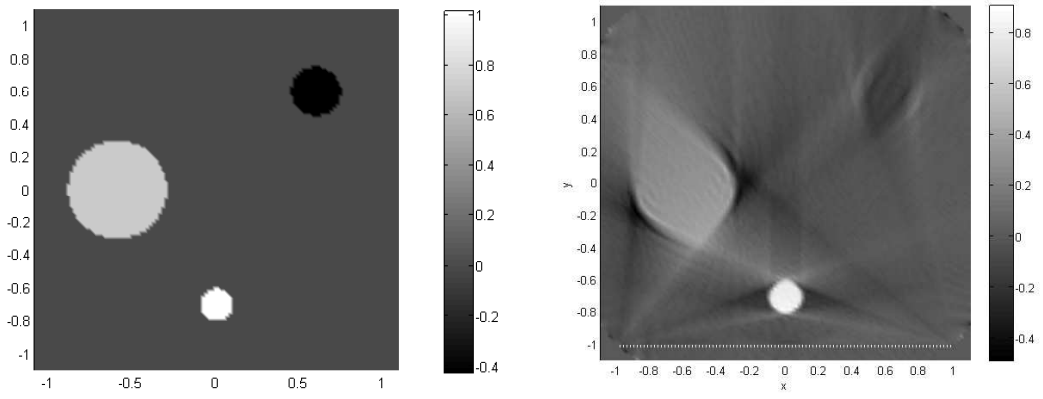

Fig. 3.6. A phantom (left) and a reconstruction (right) using method A. Only one Compton detector array at the bottom side of the square is used.

The effect of limited view data is clearly visible on the reconstruction shown in Fig. 3.6. The reconstruction was obtained via method **A**. We used one Compton detector array containing 100 detectors, which was placed on the bottom side of the unit square. The angular variable, which determines the direction of incident particles, was discretized as $\theta_i = i\Delta\theta,\ \Delta\theta = \pi/100,\ i = 1, \ldots, 99$. The phantom

used for the reconstruction is zero everywhere except at three circular inclusions, where it equals to 1, 0.6 and -0.4 respectively. We see that the boundary of the circle closest to the detector array is reconstructed well, while only parts of the boundaries of the other two circles are visible. The values of the reconstructed image differ from that of the phantom. The arcs from the circles which are recovered are exactly the ones for which the tangent lines intersect the detector array.

Next, we present reconstructions based on the methods B and C.

3.3.2. Methods B and C

In order to have complete Radon data for $s \in [-\sqrt{2}, \sqrt{2}], \phi \in (-\pi/2, \pi/2)$, we used three detector arrays placed on sides of the unit square. Methods **B** and **C** reduce the problem of inverting the 2D Cone transform to inverting the Radon transform of the function f. In order to reconstruct $f(\mathbf{y})$ from fanbeam data, we interpolated $u(\mathbf{x}, \boldsymbol{\alpha})$ from fanbeam coordinates $(\mathbf{x}, \boldsymbol{\alpha})$ to parallel beam coordinates $(s, \boldsymbol{\omega})$, where $s \in [-\sqrt{2}, \sqrt{2}]$ and $\omega \in S^1$. The convertion of coordinates is given by the relations

$$s = \mathbf{x} \cdot \boldsymbol{\omega},\ \boldsymbol{\omega} = (\sin\alpha, -\cos\alpha),\ \phi = \alpha - \pi/2 \tag{3.24}$$

Although more efficient algorithms for reconstruction from fanbeam data exist [10], we chose interpolating parallel beam data for its simplicity and independence of the geometry of detectors.

For all numerical results in this section the following setup was used. Each detector array contained 100 detectors. The discretization step for the central axes

$\boldsymbol{\beta}$ and half-angles ψ was $2\pi/128$. Moreover, $\boldsymbol{\beta}$ could not take values 0 or π, and ψ could not be equal to π.

On Figs. 3.7 and 3.8 a grayscale plot and a profile plot of a reconstruction of an analytic phantom using method **B** are shown. We used the same phantom as in the example for method **A**.

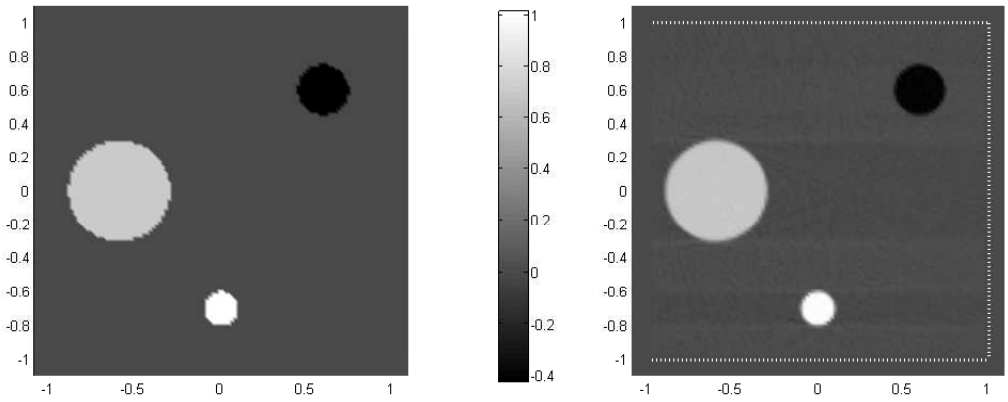

Fig. 3.7. A phantom (left) and a reconstruction (right) using method **B**. The dotted white lines represent the positions of Compton cameras.

On Fig. 3.9, a reconstruction of the phantom from Fig. 3.7 is shown. Note that the values of the reconstructed function at the detectors were cut away. Although the reconstruction with the method **B** (see Fig. 3.7) is somewhat better quality, the latter method is considerably faster.

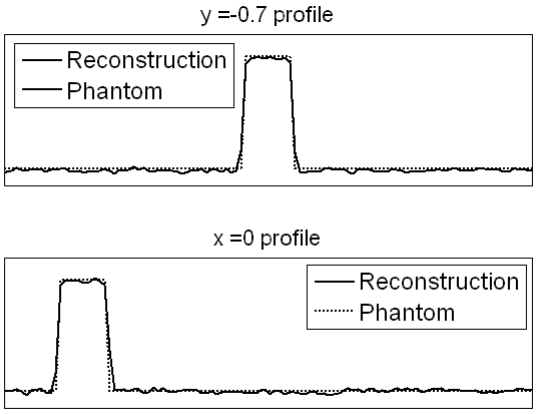

Fig. 3.8. Profile plots of the reconstruction shown of Fig. 3.7. Both profiles are through the center of the lowest disc, namely $y = -0.7$ profile (top) and $x = 0$ profile (bottom).

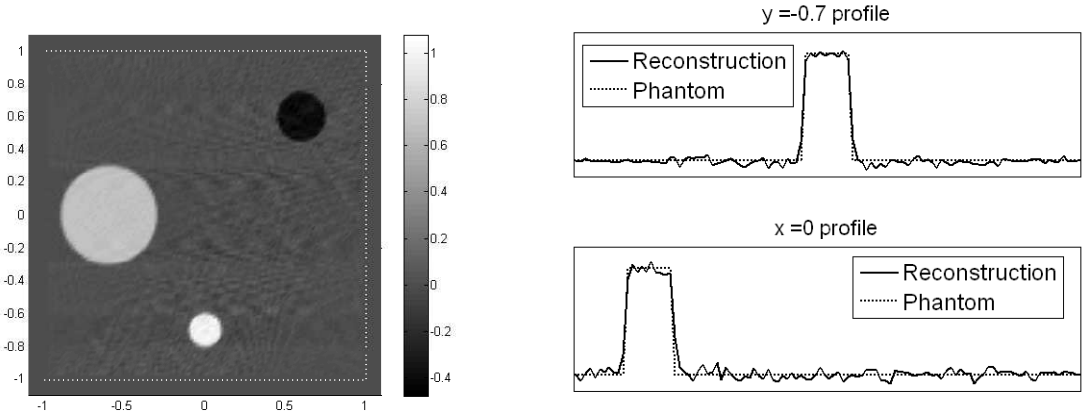

Fig. 3.9. A reconstruction of the phantom shown on Fig. 3.7 using the method described in **C**. The same setup as in Fig. 3.7 was used.

3.4. Detection of Low Emission Geometrically Small Sources

One of the missions of the Department of Homeland Security is to prevent smuggling of weapon-grade nuclear materials. It is expected that such materials, unlike those needed for a "dirty bomb", will have low emission rates and will be well shielded, so that very few gamma photons or neutrons would escape, and even less would be detected. An additional hurdle for the detection of illicit nuclear substances is the strong natural radiation background.

The most commonly used γ-ray cameras are mechanically collimated, which means that they "count" the particles coming from one direction and discard the rest. Collimation is needed because it provides crucial directional information about the particles, which can be used for mathematically reconstructing the location of radiating sources. However, mechanical collimation dramatically reduces the sensitivity of detectors. Typically, only one out of ten thousand emitted photons is detected by a collimated camera. When dealing with low emission sources one wants to capture as many of the emitted particles as possible, thus mechanical collimation is unsuitable. On the other hand, if no directional information is available, the detection of low emission sources against large noise would most likely be impossible. The ability of Compton cameras to provide some directional information without discarding any incoming particles make them of particular interest for use in such applications.

The task of detecting low emission sources in the presence of a large background would be impossible if the size of the source were large. Simulations of small enriched uranium source placed in a cargo container estimate that only about 0.1% of the

signal received by detectors might be due to ballistic (non-scattered) particles emitted from the source. The remaining 99.9% of detected particles come from natural sources and scattered source particles, and thus represents noise in the measurements. It is clear that under such conditions standard tomographic methods will fail to produce a satisfactory image. However, if smuggling is attempted, only a small volume of radioactive substance could be transported. Thus, the source will appear as almost a point singularity, and it is known in tomography that singular objects are easier to detect. Another factor which gives hope to find such sources is that all trajectories of ballistic source photons pass through the same small volume, which will increase the local density of trajectories and could conceivably allow detection. The latter argument will be discussed in detail below.

3.4.1. Source Detection by Standard Tomographic Techniques

We have attempted detecting a point source in 2D using standard tomographic reconstructions from simulated Radon data as well as from Compton camera data (see Section 3.2). We reconstructed images of radiation source density using FBP and local tomography (see Chapter I). This approach worked to some extent, although it was unreliable and started failing sooner than we expected.

There are a couple of reasons for the poor performance of these tomographic methods. First, the use of standard tomography assumes that data represents line (or cone) integrals of the source distribution function. This is the same to say that we measure the expectation of the number of hits, per unit time, from particles moving along a line (or cone). This assumption is reasonable when the source is strong,

however it fails in the case of low emission sources. Hence, the Radon transform type models are not appropriate. The second reason is that the high-pass filters required in Radon transform inversions amplify the noise. This is very significant factor, as the noise constitutes 99.9% of the data. Filtered backprojection worked to some extent, but the strong local tomorgaphy filter made source detection almost impossible.

Another drawback of the standard tomographic techniques is that the reconstructed image cannot be easily interpereted in a qualitative way, for example, it is hard to quantify the confidence level of detection.

All these considerations led us conclude that reconstruction by backprojection would be more suitable for our purposes. Indeed, in this particular problem, backprojection proved to be a considerable improvement over standard tomography. In Section 3.5 examples of reconstructions using FBP and backprojection are compared. Details of our numerical simulations are also provided there.

3.4.2. Source Detection Using Backprojection

Now we investigate why it is possible to detect *small* low emission sources by backprojection. Suppose the detectors have information about the incident direction of particles. Then backprojection "sends" the detected particles back along their incoming lines. In particular, the ballistic source particles would "cross" the source region. The background particles which scattered prior to detection will be backprojected along lines that are different from their original trajectories. Therefore, in the

considerations that follow, we assume that the trajectories of particles are straight lines. We also assume that the (random) distribution of trajectories is uniform.

In a nutshell, our argument is that *if the number of lines passing through a tiny region deviates significantly from the mean of the background, then most probably there is a source there* (see Fig. 3.10).

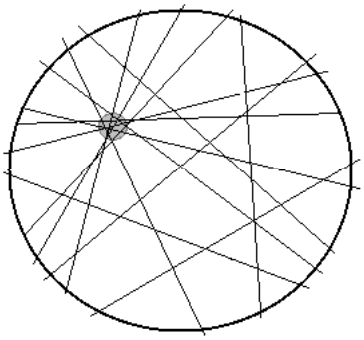

Fig. 3.10. The local density of lines is higher in the gray region, which is likely due to a source at this location.

Let us consider the trajectories of N particles in a ball B_R with radius R. We will determine the probability of k particles out of the total N contained in B_R passing through a small ball B_r with radius r. Recall that a line $L_{(\omega,s)}$ in $\mathbb{R}^2$ is defined by a vector $\boldsymbol{\omega} \in S^1$ perpendicular to it and a signed distance s from the origin (see Fig. 1.1), i.e. $L_{(\omega,s)} = \{\mathbf{x} \in \mathbb{R}^2 | \mathbf{x} \cdot \boldsymbol{\omega} = s,\ s \in \mathbb{R},\ \boldsymbol{\omega} \in S^1\}$. Thus, the lines on the plane are mapped (bijectively) to the points on the half-cylinder $S^1_+ \times (-\infty, \infty)$, where $S^1_+ = \{\boldsymbol{\omega} = (\cos\theta, \sin\theta), \theta \in [0, \pi)\}$, and these points are uniformly distributed on the half-cylinder. All lines intersecting B_R correspond to points on $S^1_+ \times (-R, R)$, and those intersecting B_r are bijectively mapped to points on $S^1_+ \times (-r, r)$. Thus,

the probability of a random line in B_R passing through B_r is

$$p = \frac{\text{area}(S^1_+ \times (-r, r))}{\text{area}(S^1_+ \times (-R, R))} = \frac{r}{R}.$$

The probability that n out of the total N lines cross B_r is given by the Binomial distribution $B(N, p)$, and is equal to $\binom{N}{n} p^n (1-p)^{N-n}$. Since we are in the situation when p is fixed by the dimension of the source and N is large, the Central Limit Theorem applies. It follows that the Binomial distribution is approximated well by the Normal distribution $N(\mu, \sigma^2)$ with mean $\mu = Np$ and standard deviation $\sigma = \sqrt{Np(1-p)}$. If it happens that the number of lines n crossing B_r deviates significantly (in comparison with σ) from the mean μ, the probability of this occurring just due to random reasons is very small. Therefore, one can be almost certain that this clustering of trajectories is the result of a radioactive source located at B_r. Now we will make these statements more precise.

Suppose that the area we are imaging is divided into N_{pix} pixels. Suppose further that n lines intersect a pixel B_r with linear dimension r. We will write $n = n_s + n_b$, where n_s and n_b are the number of source and background trajectories, respectively, which cross B_r. We will describe a method to determine whether there is a source located at this pixel.

Let us choose a threshold value k_t, such that if $n > n_t := \mu + k_t\sigma$, the pixel B_r is classified as containing a source (i.e. positive result). Otherwise there is no source at B_r. The probability of at least n_t lines crossing B_r due to random reasons is approximately $r := 0.5 \operatorname{erfc}(k_t/\sqrt{2})$. We have to take into account the possibility of such clustering of lines occurring inside some of the total N_{pix} pixels. The probability

of at least n_t lines crossing at least one of the pixels due to random reasons can be estimated by

$$\text{fp rate} = 1 - (1 - r)^{N_{pix}}. \tag{3.25}$$

Here *fp rate* stands for *false positive rate*, which is the rate of false detections [84]. We must note that the above estimate assumed independence of the events "a pixel is crossed by a line". This is obviously not true, but a correct estimate would be more complicated and would not make a significant difference in what follows next (much cruder estimates exist and can be useful sometimes).

The *true negative rate* (or specificity), is given by tn rate $= 1 -$ fp rate and represents the rate of correctly classified negative outcomes. The true negative rate is also a confidence probability that we have found a true source. We will write

$$\text{tn rate} = \text{confidence} := (1 - r)^{N_{pix}} = [1 - 0.5\ \text{erfc}(k_t/\sqrt{2})]^{N_{pix}}. \tag{3.26}$$

Obviously, this confidence probability depends only on the threshold value k_t and the number of pixels N_{pix}. One could also choose to compute the confidence based on the actual number of trajectories n crossing B_r, which is higher than n_t. Then, the estimate of confidence probability will be larger and more accurate. We should also note that the higher the number of pixels, the lower our confidence will be (see the examples below).

Now suppose there is indeed a source present at a certain pixel B_r. Suppose also that we have an estimate of the number of detected source particles. We can write $n_s = k_s\sigma$. Here k_s is a positive constant and σ is the standard deviation of the distribution of the trajectories. Recall that σ is determined by the total number

of detected particles, which is known, and the ratios of the source dimension to the dimension of the imaged region, for which we would, in practice, have a crude idea. Then, the source would not be detected if the total number of lines crossing B_r, n, is smaller than the threshold n_t, i.e. if

$$n = n_b + n_s < n_t = \mu + k_t\sigma = \mu + (k_t - k_s)\sigma + k_s\sigma, \quad \text{where } n_s = k_s\sigma \tag{3.27}$$

This implies that $n_b < \mu + (k_t - k_s)\sigma$, where n_b is the number of background particles crossing B_r. The probability of this happening is given by

$$\text{fn rate} = 0.5 \text{ erfc}\left(\frac{k_s - k_t}{\sqrt{2}}\right). \tag{3.28}$$

In other words, we have determined the *false negative rate* (fn rate) of detection. The sensitivity, or *true positive rate* (tp rate), is given by

$$\text{tp rate} = 1 - \text{fn rate}.$$

Note that the sensitivity of the method depends on the threshold value k_t as well as on our estimate of detected source particles.

Examples

In practice, a nuclear source with shielding could have a radius of the order of several centimeters, while a vehicle or a cargo container has size in the order of meters. Thus, we will assume that

$$R = 1, \ r = 0.01 \quad \Rightarrow p = 10^{-2}. \tag{3.29}$$

1. Suppose that we image an area in two dimensions and that it is divided into

pixels with linear dimension equal to $1/100^{th}$ of the linear dimension of the whole region. Then the probability of one line crossing a certain pixel is $p \approx 100^{-1}$ and the total number of pixels is $N_{pix} = 100^2$. Suppose further that we have detected a total of $N = 10^6$ particles. We can determine the standard deviation of the trajectory distribution to be $\sigma = \sqrt{Np(1-p)} \approx 99.5$. As we mentioned in Section 3.4, the expected SNR is about $s = 0.001$. This gives us the approximate number of detected ballistic source particles,

$$n_s = 10^3 = k_s.\sigma,$$

with $k_s \approx 10$. Let us set the threshold to $k_t = 5$.

If we detect a source at a certain pixel, the true negative rate will be determined by (see equation (3.26))

$$[1 - 0.5 \text{ erfc}(5/\sqrt{2})]^{100^2} = 0.997$$

In other words we can be certain at least 99.7% that the source is a true one.

The probability that we fail to detect a source is given by (3.28)

$$0.5 \text{ erfc}\left(\frac{10-5}{\sqrt{2}}\right) \approx 3.10^{-7}$$

2. In three dimensions, if the same resolution is used, the number of pixels is $N_{pix} = 100^3$. Suppose we have detected the same level of particles as in the example above, i.e. $N = 10^6$. We note that the probability $p = 100^{-1}$ is the same in two and three dimensions. Thus $\sigma \approx 99.5$, and the total number of detected ballistic source particles is $n_s \approx 10\sigma$, as above.

Let us set the threshold to $k_t = 5$ as in 2D. Then the true negative rate will be approximately 0.75, i.e. our confidence that we have found the source will be at least 75%. This drop in confidence in three dimensions is due to the higher number of pixels, 100^3, compared to the number of pixles in 2D, which is 100^2 (see equation (3.26)). The probability that we fail to detect a source will be as in 2D: fn rate $= 3.10^{-7}$.

If we choose a threshold $k_t = 6$, then we can be confident that a detected source is a true one 99.9%. The false negative rate will be approximately 3.10^{-5}.

From the above discussion we see that in order to find sources reliably, we need to detect $n_s > k_t\sigma$ particles from the source, where the threshold k_t could be quite large. We claim that for any $k_t > 0$ it is possible to detect sources, provided N is sufficiently large. Indeed, if a signal to noise ratio s is known, and if $p \ll 1$, the number of detected source particles will be $n_s \approx sN$. Then, we requre the following inequality to be satisfied

$$sN > k_t\sigma = k_t\sqrt{Np(1-p)}. \tag{3.30}$$

Since the left hand side of (3.30) grows linearly with N, while the right hand side grows only as $\sqrt{N}$, the inequality will be satisfied for a sufficiently large N.

In practice, as we saw in the examples above, in order to detect $n_s \approx 10\sigma$ particles from the source, we would need a total of 10^6 detected particles. This number is realistic and could be achieved in at most a few minutes of detection. Exact times will depend on the type of particles (γ-photons or neutrons of some specific energies) and of the type and size of detector arrays used.

3.5. Numerical Examples

For the numerical examples shown next, we simulated a point source placed inside the unit square in two dimensions. The source emitted photons uniformly in all directions and the background was uniformly distributed. The particles were detected by three detector arrays placed on three sides of the square. The use of three detector arrays was deemed appropriate, because this configuration resembles a "gate", through which vehicles and cargo would pass. We simulated mechanically collimated detectors, which render Radon type data, and also Compton camera type detectors, which provide cone data. Each of the three detector arrays contained 100 detectors. The angular variable, which determines the direction of incident particles, was discretized as $\theta_i = i\Delta\theta,\ \Delta\theta = \pi/200,\ i = 1, \ldots, 199$ (angular bins).

3.5.1. Reconstruction by Standard Tomographic Techniques

First, we experimented with standard tomographic inversions. We assumed that the collected data represented line (or cone) integrals of the source distribution function. This means that, in the case of Radon-type detectors, we interpolated the fanbeam data $u(\mathbf{x}, \boldsymbol{\alpha})$ to parallel beam coordinates $(s, \boldsymbol{\omega})$, exactly as described in Section 3.3. On Fig. 3.11 an example of a reconstruction from (Radon) data collected by three collimated detectors is shown. The number of detected source particles was 1039, versus 10^6 background particles. The source was located at $(0.203, -0.101)$. The reconsruction peaks not only at the correct source location but also at a number of other places. Even worse, some of those incorrect peaks are higher than the peak

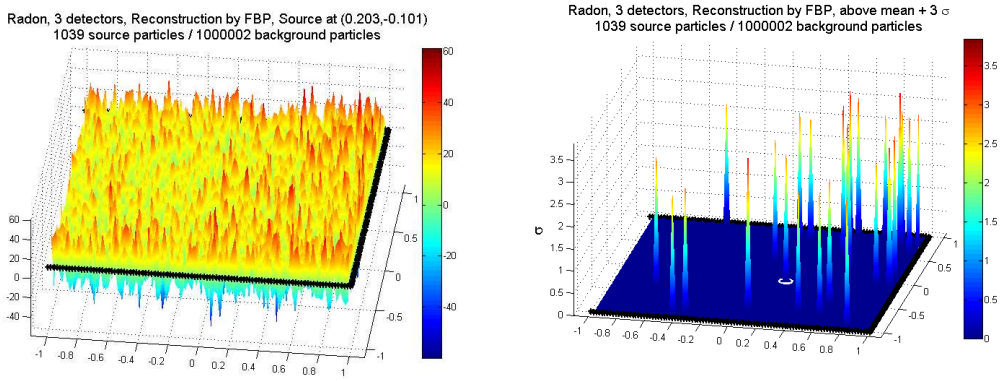

Fig. 3.11. Filtered backprojection reconstruction from Radon data (left). The black lines represent the detectors. The right picture shows only the peaks that deviate more than 3 standard deviations from the mean. The correct source location is circled.

at the source. The probabilistic considerations given in Section 3.4 suggest that the source should be detectable. When Compton camera type detectors were used, data from cone integrals was converted to line integrals data by method C, described in Section 3.2.2. After this conversion we used filtered backprojection to reconstruct the source distribution. On Fig. 3.12 a reconstruction from data obtained by three Compton cameras is shown. The number of detected source particles was 1039, versus 10^6 background particles. The source was located at $(0.203, -0.101)$. The result is similar to the previous example.

On the next figure, Fig. 3.13, we show a FBP reconstruction from Radon data. We detected 994 particles from a source located at $(-0.503, -0.101)$ and 10^6 uniformly distributed background particles. The particles were detected by Radon-type

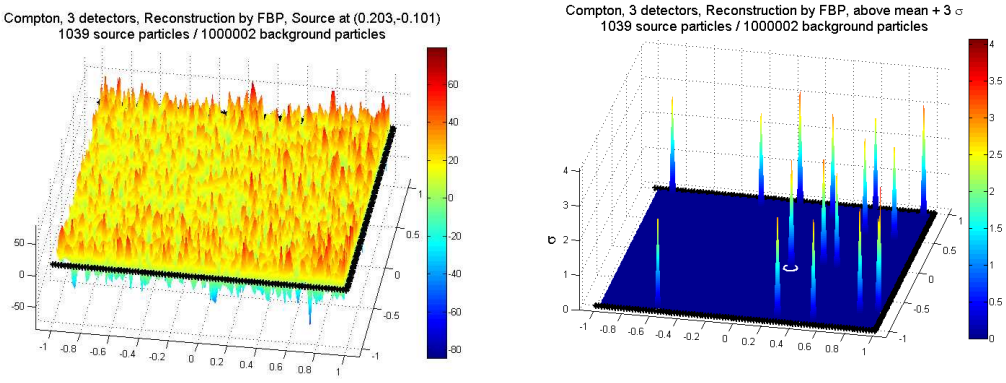

Fig. 3.12. Filtered backprojection reconstruction from Compton data (left). The black lines represent the detectors. The right picture shows only the peaks that deviate more than 3 standard deviations from the mean. The correct source location is circled.

detectors placed on three sides of a square. On Fig. 3.13 we show only peaks in the reconstruction which deviate more than 4.5 standard deviations from the mean (i.e. we have set up the threshold to be $k_t = 4.5$). The true source is detected and there are no false detections. Fig. 3.14 shows a successful detection of a source from Compton camera data. For this simulation we used the same SNR, 0.001, but a higher number of background particles - $1,500,000$. The source was located at $(0.203, -0.101)$. The graph shows only the correct peak with value more than $k_t 4$ standard deviations above the mean.

In summary, standard tomographic techniques fail to reliably detect sources at the levels of SNR and total number of particles for which our probabilistic considerations show that sources should be detectable. These techniques work well with

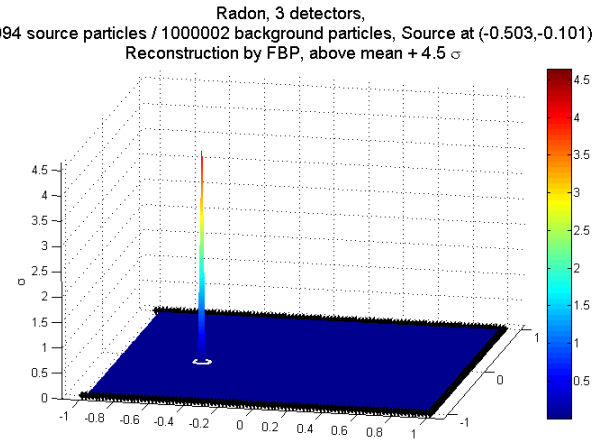

Fig. 3.13. FBP reconstruction from Radon data, with 994 source particles and 10^6 background particles. The graph shows only the peaks that deviate more than 4.5 standard deviations from the mean. The black lines represent the detector arrays. The position of the correct source location is circled.

higher SNRs, e.g. $s = 0.01$ [85].

3.5.2. Reconstruction by Backprojection

Reconstructions by backprojection (see equation (1.11)) agree with our expectations described in Section 3.4. As a first example, on Fig. 3.15 we present a typical histogram of the number of lines crossing a pixel in the 2D unit square with a uniform grid of 100^2 pixels. The particles where detected by 4 collimated detectors placed on the sides of the square. The signal to noise ratio was 0.001 and the number of background particles particles was $1,333,336$. The detected particles were backprojected, and after that the historgam of the backprojection reconstruction was

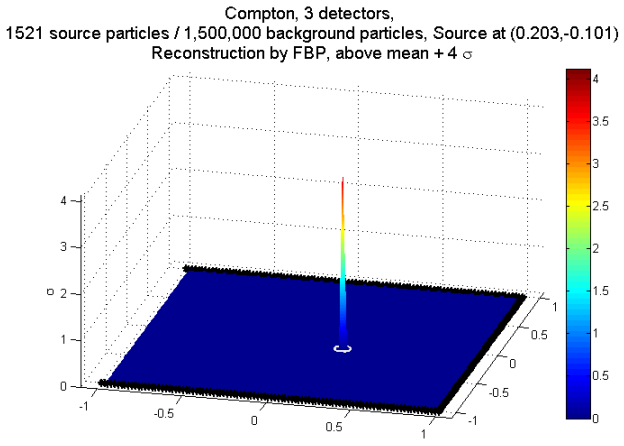

Fig. 3.14. FBP reconstruction from Compton data, with 1521 source particles and $1,500,000$ background particles. The graph show only the peaks that deviate more than 4 standard deviations from the mean. The black lines represent the detectors. The position of the correct source location is circled.

plotted. On the far right of the histogram one sees the contribution from the source, which deviates more than 10 standard deviations from the mean. This result complies with our predictions.

We first illustrate our processing of data on a typical example of Radon type data. In fact, the data used for this backprojection reconstruction was the same used for the unsuccessful FBP reconstruction shown on Fig. 3.11. We backprojected the detected particles from each of the three detector arrays separately and then added the results. The first picture on Fig. 3.16 shows a typical backprojection reconstruction. There is a drop of the values at the side of the square, due to the missing

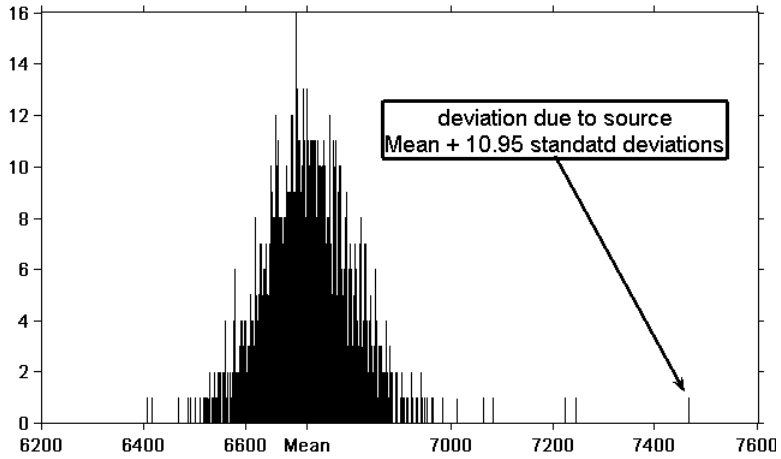

Fig. 3.15. Histogram of a backprojection reconstruction from data collected by four collimated detectors, placed on the sides of the unit square. The data simulated a source at the point $(0.202, -0.01)$, with SNR 0.1% and total number of particles slightly more than 10^6. The number of lines due to the source deviates more than 10 standard deviations from the mean.

detector. Thus, effectively, there is a non-uniform distribution of the background. However, this change in the non-uniformity is smooth, so we can assume that the *local distribution of trajectories is uniform*. In order to apply our probabilistic arguments described in Section 3.5.2 we estimated the *local* mean and standard deviation at every point. This was done by analysis of the data on a 7×7 grid surrounding each pixel. Then, the standard deviation from the local mean at every point was computed and plotted (see bottom left plot on Fig. 3.16). The confidence probability

was calculated by

$$\text{confidence} = (1 - 0.5 * \text{erfc}(k_{source}/\sqrt{(2)}))^{100^2},$$

where k_{source} is the number standard deviations above the mean at the highest peak. The calculated confidence level for this example was 99.99%.

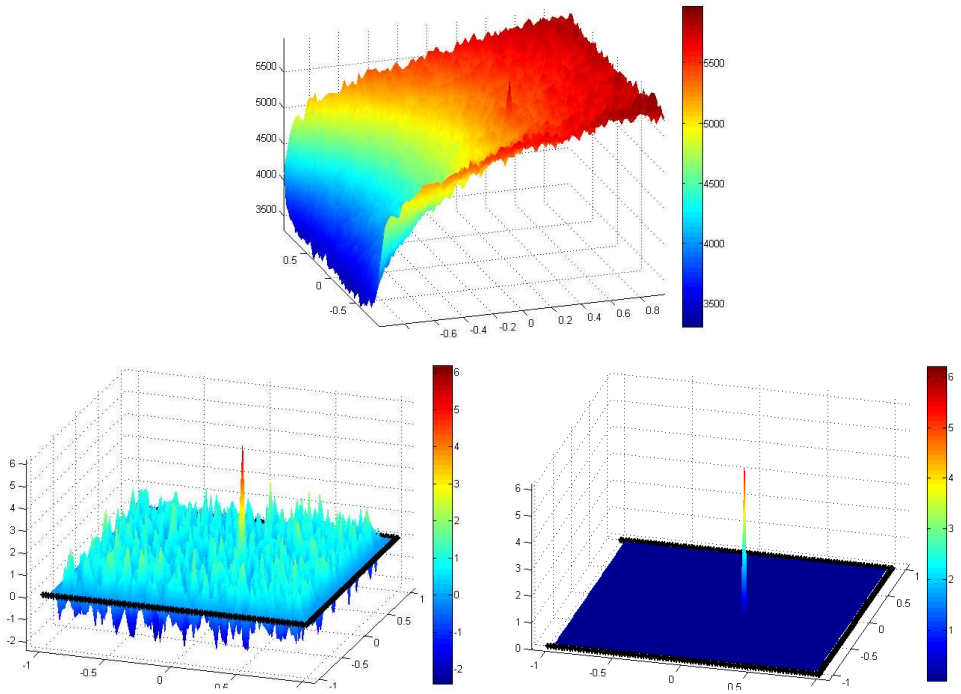

Fig. 3.16. Backprojection reconstruction from Radon data (top). Number of (local) standard deviations above the (local) mean at each pixel is shown on bottom left. Peaks that deviate more than 3.5 (local) standard deviations from the mean is shown on bottom right. Source located at $(0.203, 0.101)$ is detected with confidence 99.99%.

For the reconstructions from Compton camera data, we used method C to con-

vert cone data into Radon data. Afterwards, the procedures used for Radon data (described above) were employed. The backprojection reconstruction from the same data used in Fig. 3.12 is shown on Fig. 3.17. The results are similar to the reconstruction from Radon data, only the confidence probability is slightly lower - 97.56%.

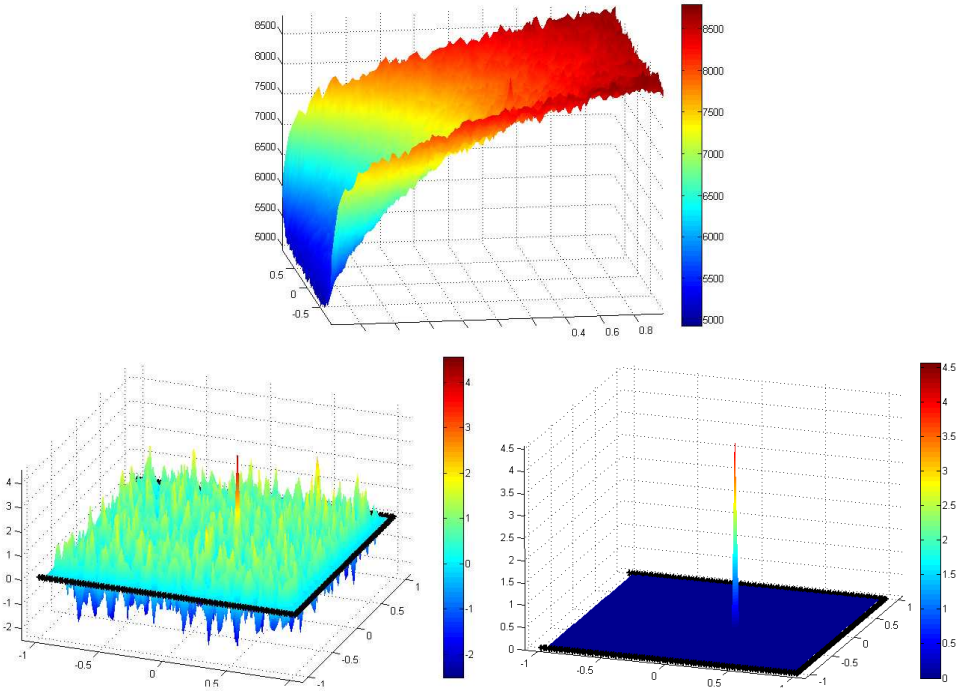

Fig. 3.17. Backprojection reconstruction from Compton data (top). Number of (local) standard deviations above the (local) mean at each pixel is shown on bottom left. Peaks that deviate more than 3.5 (local) standard deviations from the mean is shown on bottom right. Source located at $(0.203, 0.101)$ is detected with confidence 97.56%.

A couple of more examples of reconstructions from Compton camera data are

provided below. The number of source particles was 986 versus 10^6 background particles. The source on Fig. 3.18 is close to one of the detectors and is difficult to detect. The local mean and σ were estimated on a larger local grid, 11×11 pixels, which raised the confidence level. We believe that larger local grids allow for more precise estimation of the mean, thus raising the confidence level for the detected source. However, if the local grids are too large, the distribution of trajectories in a local grid would not be close to uniform anymore, and this will hamper our ability for detection. The source on Fig. 3.19 is close to the side of the square on which a detector is missing. The number of source particles was 994 versus 10^6 background particles. The local mean and standard deviation were estimated on 7×7 local grids. The confidence level is 96.7%.

Finally, we present an example of a successful detection of several sources (Fig. 3.20). Three sources were placed inside the imaged area, and three Compton-type detector arrays were used to detect particles. The total number of detected source particles was 3061, each of the sources having roughly the same contribution. Additional 10^6 random background particles were detected. We have set up the threshold to $k_t = 4.3\sigma$, which gives confidence of detection at least 91.97%. Note that this confidence probability is estimated base on the threshold, and not on the actual number of deviations at each peak, as in the previous examples. The local grids we used for estimating the local means and standard deviations had dimensions 11×11.

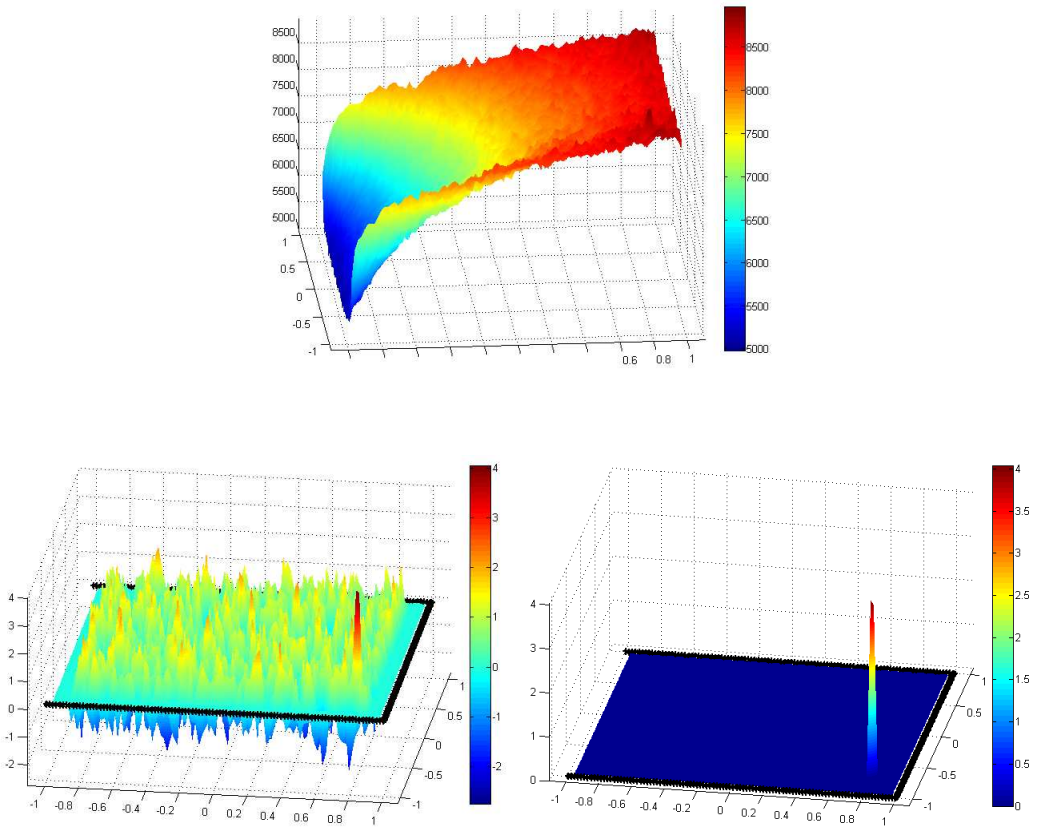

Fig. 3.18. Backprojection reconstruction from Compton data (top). Detected source particles - 986, background particles - $1,000,002$, source location - $(0.803, -0.7041)$. Number of (local) standard deviations above the (local) mean at each pixel is shown on bottom left. The local mean and σ were computed on a 11×11 local grid. Peaks that deviate more than 3.5 (local) standard deviations from the mean is shown on bottom right. Source located at $(0.803, 0.7041)$ is detected with confidence 76.03%.

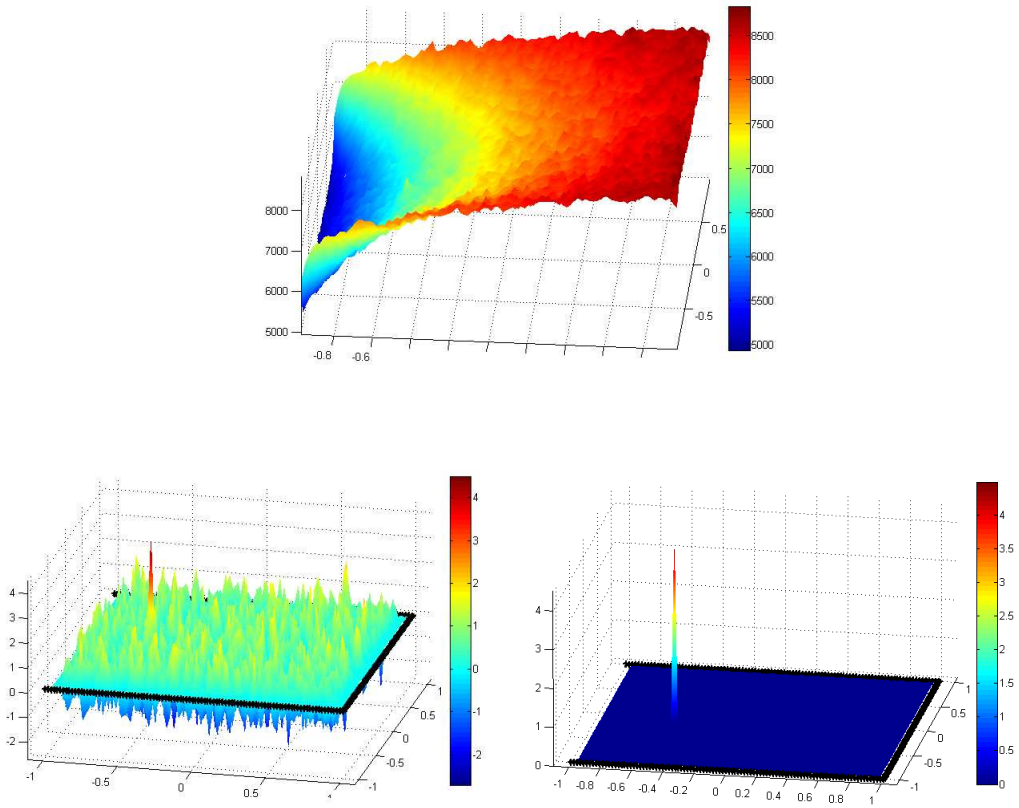

Fig. 3.19. Backprojection reconstruction from Compton data (top). Detected source particles - 994, background particles - $1,000,002$, source location - $(-0.503, -0.101)$. Number of (local) standard deviations above the (local) mean at each pixel is shown on bottom left. Peaks that deviate more than 3.5 (local) standard deviations from the mean is shown on bottom right. Source located at $(0.503, 0.101)$ is detected with confidence 96.7%.

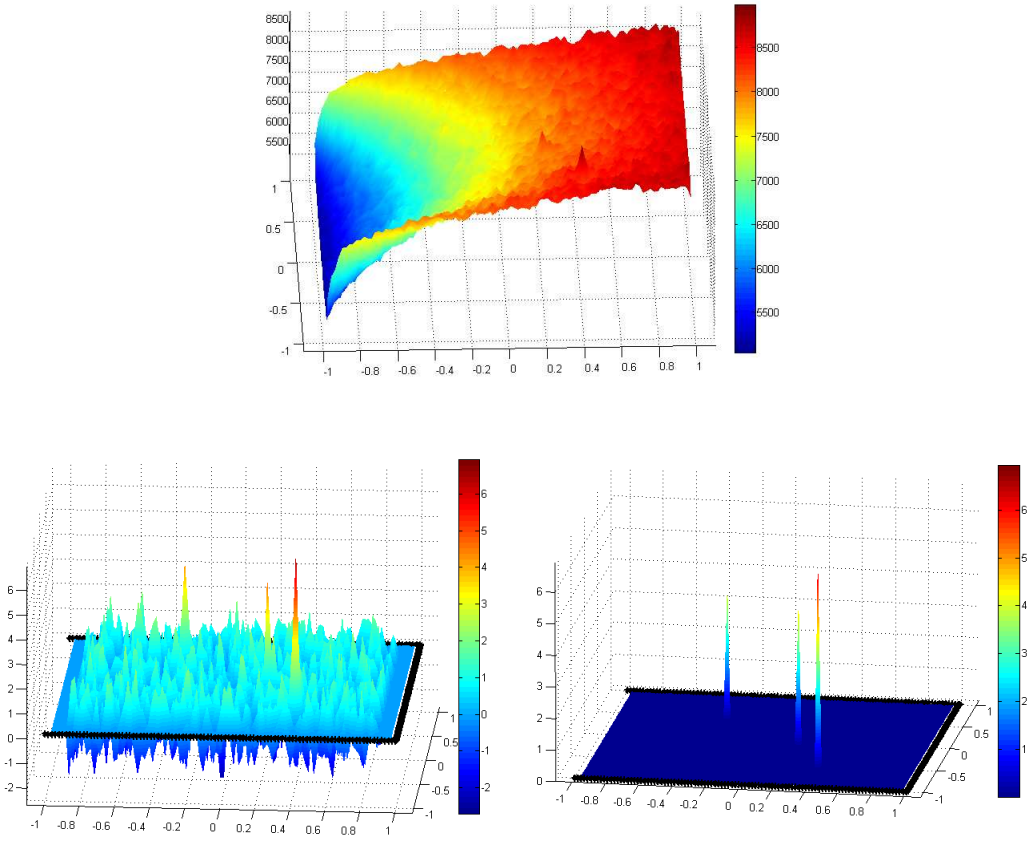

Fig. 3.20. Backprojection reconstruction from Compton data (top). Detected source particles - 3061, background particles - $1,000,002$, sources located at (0.203,-0.101), (-0.301,0.4013) and (0.403,-0.601). Number of (local) standard deviations above the (local) mean at each pixel is shown on bottom left. Peaks that deviate more than 4.3 (local) standard deviations from the mean is shown on bottom right. Confidence of detection based on the threshold $k_t = 4.3$ is 91.97%.

3.6. Remarks

In this chapter we investigated three methods of inversion of the two-dimensional cone transform, which arises in Compton camera imaging. All of these methods use the complete set of available data, which is highly desirable in detection of low emission sources. One of the presented methods, method C (see Section 3.2.2), reconstructs the correct image away from the detectors, and, in addition, is simple and numerically efficient. In 3D, the data set provided by Compton cameras is five dimensional and computationally efficient methods are needed. Thus, an analog of method C in three dimensions could be highly valuable.

In Sections 3.4 and 3.5 we considered the problem of detection of low emission geometrically small sources in the presence of strong natural radiation background. We presented a method for detection and discussed its application to a homeland security project aimed at preventing the smuggling of nuclear substances. Examples of source detection from numerically simulated data using Compton camera detectors in 2D were shown. We described a technique for source detection when the background is not uniform. Testing of our algorithm with more realistic data is in process.

CHAPTER IV

CONCLUSION

In the first part of this dissertation we described the time reversal method, as applied to thermoacoustic tomography. We presented a thorough numerical study of the method, which included reconstructions with variable trapping and non-trapping sound speed, non-zero initial velocity, phantoms partially supported outside the imaged area as well as reconstructions with averaged sound speed. We provided error estimates for the time reversal approximation, thus justifying the time reversal method in any dimension under a non-trapping condition on the sound speed. Numerical examples agree with the error estimates in the cases of non-trapping speeds and show that time reversal works even when the sound speed is trapping. The stability of the time reversal method with respect to errors in measured data was also investigated.

The second part of this dissertation considered a problem from emission tomography, namely Compton camera imaging and the resulting cone transform. We presented three methods for inversion of the two-dimensional cone transform and provided examples of reconstructions based on these methods. Next, we described the problem of detection of geometrically small sources in the presence of a large background radiation and the use of Compton cameras for this purpose. We presented an efficient method for detection of such sources and provided numerical examples which demonstrate the method.

REFERENCES

[1] R. Gordon and G. T. Herman, "Reconstruction of pictures from their projections," *Communications of the ACM*, vol. 14, no. 12, pp. 759–768, 1971.

[2] M. Xu and L. V. Wang, "Photoacoustic imaging in biomedicine," *Review of Scientific Instruments*, vol. 77, art. no. 041101, 2006.

[3] L. V. Wang and H. Wu, *Biomedical Optics: Principles and Imaging*. Hoboken, NJ: John Wiley & Sons, Inc., 2007.

[4] D. Finch and Rakesh, "The spherical mean value operator with centers on a sphere," *Inverse Problems*, vol. 23, no. 6, pp. S37–S49, 2007.

[5] P. Kuchment and L. Kunyansky, "Mathematics of thermoacoustic tomography," *European J. Appl. Math.*, vol. 19, no. 2, pp. 191–224, 2008.

[6] M. Agranovsky, P. Kuchment, and L. Kunyansky, "On reconstruction formulas and algorithms for the thermoacoustic and photoacoustic tomography," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 89–101, Boca Raton, FL: CRC Press, 2009.

[7] D. Finch and Rakesh, "Recovering a function from its spherical mean values in two and three dimensions.," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 77–87, Boca Raton, FL: CRC Press, 2009.

[8] S. Patch and O. Scherzer, "Guest editors' introduction: Photo- and thermoacoustic imaging," *Inverse Problems*, vol. 23, no. 6, pp. S1–S10, 2007.

[9] M. Haltmeier, T. Schuster, and O. Scherzer, "Filtered backprojection for thermoacoustic computed tomography in spherical geometry," *Math. Methods Appl. Sci.*, vol. 28, no. 16, pp. 1919–1937, 2005.

[10] F. Natterer, *The Mathematics of Computerized Tomography*. Philadelphia, PA: Society for Industrial Mathematics, 2001.

[11] S. Helgason, *The Radon Transform*, vol. 5 of *Progress in Mathematics*. Boston, MA: Birkhäuser Boston Inc., second ed., 1999.

[12] Muskhelishvili, N I, *Singulyarnye Integralnye Uravneniya. Granichnye Zadachi Teorii Funktsii i Nekotorye Ikh Prilozheniya k Matematicheskoi Fizike. [Singular Integral Equations: Boundary Problems of Function Theory and Their Application to Mathematical Physics]*. Izdat. "Nauka", Moscow, augmented ed., 1968.

[13] Y. Hristova, P. Kuchment, and L. Nguyen, "Reconstruction and time reversal in thermoacoustic tomography in acoustically homogeneous and inhomogeneous media," *Inverse Problems*, vol. 24, no. 5, art. no. 055006, 2008.

[14] Y. Hristova, "Time reversal in thermoacoustic tomographyan error estimate," *Inverse Problems*, vol. 25, no. 5, art. no. 055008, 2009.

[15] L. V. Wang, "Microwave-induced acoustic (thermoacoustic) tomography," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 339–347, Boca Raton, FL: CRC Press, 2009.

[16] A. C. Tam, "Applications of photoacoustic sensing techniques," *Reviews of Modern Physics*, vol. 58, no. 2, pp. 381–431, 1986.

[17] B. T. Cox and P. C. Beard, "Modeling photoacoustic propagation in tissue using k-space techniques," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 25–34, Boca Raton, FL: CRC Press, 2009.

[18] M. Xu and L. V. Wang, "Time-domain reconstruction for thermoacoustic tomography in a spherical geometry," *IEEE Transactions on Medical Imaging*, vol. 21, no. 7, pp. 814–822, 2002.

[19] L. C. Evans, *Partial Differential Equations*, vol. 19 of *Graduate Studies in Mathematics*. Providence, RI: American Mathematical Society, 1998.

[20] D. Finch, S. Patch, and Rakesh, "Determining a function from its mean values over a family of spheres," *SIAM Journal on Mathematical Analysis*, vol. 35, pp. 1213–1240, 2004.

[21] D. Finch, M. Haltmeier, and Rakesh, "Inversion of spherical means and the wave equation in even dimensions," *SIAM Journal on Applied Mathematics*, vol. 68, pp. 392–412, 2007.

[22] L. A. Kunyansky, "Explicit inversion formulae for the spherical mean Radon transform," *Inverse Problems*, vol. 23, no. 1, pp. 373–383, 2007.

[23] L. Nguyen, "A family of inversion formulas in thermoacoustic tomography," *Inverse Problems and Imaging*, vol. 3, no. 4, pp. 649 – 675, 2009.

[24] M. Agranovsky and P. Kuchment, "Uniqueness of reconstruction and an inversion procedure for thermoacoustic and photoacoustic tomography with variable sound speed," *Inverse Problems*, vol. 23, no. 5, pp. 2089-2102, 2007.

[25] L. A. Kunyansky, "A series solution and a fast algorithm for the inversion of the spherical mean Radon transform," *Inverse Problems*, vol. 23, no. 6, pp. S11-S20, 2007.

[26] P. Burgholzer, G. J. Matt, M. Haltmeier, and G. Paltauf, "Exact and approximative imaging methods for photoacoustic tomography using an arbitrary detection surface," *Physical Review E*, vol. 75, no. 4, art. no. 46706, 2007.

[27] P. Stefanov and G. Uhlmann, "Thermoacoustic tomography with variable sound speed," *Inverse Problems*, vol. 25, no. 7, art. no. 075011, 2009.

[28] M. Fink and C. Prada, "Acoustic time-reversal mirrors," *Inverse Problems*, vol. 17, no. 1, pp. 1–38, 2001.

[29] M. Fink, "Time-reversal acoustics," *Journal of Physics: Conference Series*, vol. 118, no. 1, art. no. 012001, 2008.

[30] P. Blomgren, G. Papanicolaou, and H. Zhao, "Super-resolution in time-reversal acoustics," *The Journal of the Acoustical Society of America*, vol. 111, pp. 230-248, 2002.

[31] M. A. Anastasio, J. Zhang, X. Pan, Y. Zou, G. Ku, and L. V. Wang, "Half-time image reconstruction in thermoacoustic tomography," *IEEE Transactions on Medical Imaging*, vol. 24, no. 2, pp. 199–210, 2005.

[32] Y. Xu and L. V. Wang, "Effects of acoustic heterogeneity in breast thermoacoustic tomography," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 50, no. 9, pp. 1134–1146, 2003.

[33] Y. Xu, L. V. Wang, G. Ambartsoumian, and P. Kuchment, "Reconstructions in limited-view thermoacoustic tomography," *Medical Physics*, vol. 31, pp. 724-733, 2004.

[34] G. Paltauf, R. Nuster, M. Haltmeier, and P. Burgholzer, "Limited view photoacoustic tomography with line detectors," *Inverse Problems*, vol. 23, pp. S81–S94, 2007.

[35] L. Nirenberg, *Lectures on linear partial differential equations.* Providence, R.I.: American Mathematical Society, 1973. Expository Lectures from the CBMS Regional Conference held at the Texas Technological University, Lubbock, Tex., May 22–26, 1972, Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics, No. 17.

[36] M. E. Taylor, *Pseudodifferential Operators*, vol. 34 of *Princeton Mathematical Series.* Princeton, NJ: Princeton University Press, 1981.

[37] Y. V. Egorov and M. A. Shubin, "Linear partial differential equations. Foundations of the classical theory," in *Partial Differential Equations, I*, vol. 30 of *Encyclopaedia Math. Sci.*, pp. 1–259, Berlin: Springer, 1992.

[38] B. R. Vainberg, "The short-wave asymptotic behavior of the solutions of stationary problems, and the asymptotic behavior as $t \to \infty$ of the solutions of nonstationary problems," *Uspehi Mat. Nauk*, vol. 30, no. 2(182), pp. 3–55, 1975. English translation: Russian Math. Surveys 30 (1975), no. 2, 1–58.

[39] J. V. Ralston, "Solutions of the wave equation with localized energy," *Comm.*

Pure Appl. Math., vol. 22, pp. 807–823, 1969.

[40] R. Pauen, "Non-trapping Conditions and Local Energy Decay for Hyperbolic Problems," vol. 132 of *Konstanzer Schriften in Mathematik und Informatik*, http://www.ub.uni-konstanz.de/kops/volltexte/2006/2146/: preprint, 2006.

[41] R. Courant and D. Hilbert, *Methods of Mathematical Physics. Vol. II: Partial Differential Equations.* (Vol. II by R. Courant.), New York: Interscience Publishers (a division of John Wiley & Sons), 1962.

[42] A. K. Louis and E. T. Quinto, "Local tomographic methods in Sonar," in *Surveys on Solution Methods for Inverse Problems* (D. Colton, Ed), pp. 147–154, Austria: Springer Verlag Wien, 2000.

[43] E. T. Quinto, "Singularities of the X-ray transform and limited data tomography in R^2 and R^3," *SIAM Journal on Mathematical Analysis*, vol. 24, no. 5, pp. 1215–1225, 1993.

[44] P. Kuchment, K. Lancaster, and L. Mogilevskaya, "On local tomography," *Inverse Problems*, vol. 11, no. 3, pp. 571–589, 1995.

[45] Y. Xu, L. V. Wang, G. Ambartsoumian, and P. Kuchment, "Limited view thermoacoustic tomography," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 61–73, Boca Raton, FL: CRC Press, 2009.

[46] L. Nguyen, "Singularities and instability in thermoacoustic tomography," preprint, 2009.

[47] X. Jin and L. V. Wang, "Thermoacoustic tomography with correction for acoustic speed variations," *Physics in Medicine and Biology*, vol. 51, pp. 6437-6448, 2006.

[48] X. Jin and L. V. Wang, "Thermoacoustic reconstruction in acoustically heterogeneous media with the aid of ultrasound tomography," in *Photoacoustic Imaging and Spectroscopy* (L. H. Wang, Ed.), pp. 473–480, Boca Raton, FL: CRC Press, 2009.

[49] Y. Xu and L. V. Wang, "Effects of acoustic heterogeneity in breast thermoacoustic tomography," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 50, no. 9, pp. 1134–1146, 2003.

[50] J. L. Lions and E. Magenes, *Non-homogeneous Boundary Value Problems and Applications. Vol. 1.* New York: Springer-Verlag, 1972.

[51] J. Ralston, "Gaussian beams and the propagation of singularities," in *Studies in Partial Differential Equations*, vol. 23 of *MAA Stud. Math.*, pp. 206–248, Washington, DC: Math. Assoc. America, 1982.

[52] N. Burq, "Décroissance de l'énergie locale de l'équation des ondes pour le problème extérieur et absence de résonance au voisinage du réel," *Acta Math.*, vol. 180, no. 1, pp. 1–29, 1998.

[53] J. L. Lions and E. Magenes, *Non-homogeneous Boundary Value Problems and Applications. Vol. II.* New York: Springer-Verlag, 1972.

[54] R. W. Todd, J. M. Nightingale, and D. B. Everett, "A proposed gamma camera," *Nature*, vol. 251, pp. 132–134, 1974.

[55] M. Singh, "An electronically collimated gamma camera for single photon emission computed tomography. Part I: Theoretical considerations and design criteria," *Medical Physics*, vol. 10, pp. 421-427, 1983.

[56] M. Singh and D. Doria, "An electronically collimated gamma camera for single photon emission computed tomography. Part II: Image reconstruction and preliminary experimental measurements," *Medical Physics*, vol. 10, pp. 428–435, 1983.

[57] V. Schönfelder, H. Aarts, K. Bennett, H. De Boer, J. Clear, W. Collmar, A. Connors, A. Deerenberg, R. Diehl, A. von Dordrecht, et al., "Instrument description and performance of the imaging gamma-ray telescope COMPTEL aboard the Compton Gamma-Ray Observatory," *Astrophysical Journal*, vol. 409, pp. 492–492, 1993.

[58] D. L. Gunter, "Filtered back-projection algorithm for Compton telescopes," 2008. US Patent 7,345,283.

[59] S. Watanabe, T. Tanaka, K. Nakazawa, T. Mitani, K. Oonuki, T. Takahashi, T. Takashima, H. Tajima, Y. Fukazawa, M. Nomachi, et al., "A Si/CdTe semiconductor Compton camera," *IEEE Transactions on Nuclear Science*, vol. 52, no. 5 Part 3, pp. 2045–2051, 2005.

[60] G. J. Royle and R. D. Speller, "Compton scatter imaging of a nuclear industry

site," in *IEEE Nuclear Science Symposium, 1997*, pp. 365–368, 1997.

[61] G. J. Royle and R. D. Speller, "A flexible geometry Compton camera for industrial gamma ray imaging," in *IEEE Nuclear Science Symposium. Conference Record.*, vol. 2, pp. 821–824, 1996.

[62] N. H. Clinthorne, C. Y. Ng, C. H. Hua, J. E. Gormley, J. W. Leblanc, S. J. Wilderman, and W. L. Rogers, "Theoretical performance comparison of a Compton-scatter apertureand parallel-hole collimator," in *IEEE Nuclear Science Symposium. Conference Record.*, vol. 2, pp. 788–792 1996.

[63] J. W. LeBlanc, N. H. Clinthorne, C. H. Hua, E. Nygard, W. L. Rogers, D. K. Wehe, P. Weilhammer, and S. J. Wilderman, "C-SPRINT: a prototype Compton camera system for low energy gammaray imaging," *IEEE Transactions on Nuclear Science*, vol. 45, no. 3 Part 1, pp. 943–949, 1998.

[64] A. W. Lackie, K. L. Matthews, B. M. Smith, W. Hill, W. H. Wang, and M. L. Cherry, "A directional algorithm for an electronically-collimated gamma-ray detector," in *IEEE Nuclear Science Symposium Conference Record*, vol. 1, pp. 264–269, 2006.

[65] W. H. Hill Jr and K. L. Matthews II, "Experimental verification of a hand held electronically-collimated radiation detector," in *IEEE Nuclear Science Symposium Conference Record*, vol. 5, pp. 3792–3797, 2007.

[66] M. J. Cree and P. J. Bones, "Towards direct reconstruction from a gamma camera based on Compton scattering," *IEEE Transactions on Medical Imaging*,

vol. 13, no. 2, pp. 398–407, 1994.

[67] R. Basko, G. L. Zeng, and G. T. Gullberg, "Application of spherical harmonics to image reconstruction for the Compton camera," *Physics in Medicine and Biology*, vol. 43, pp. 887–894, 1998.

[68] T. T. Truong, M. K. Nguyen, and H. Zaidi, "The mathematical foundations of 3D Compton scatter emission imaging," *International Journal of Biomedical Imaging*, vol. 2007, art. no. 92780, 2007.

[69] M. K. Nguyen, T. T. Truong, and P. Grangeat, "Radon transforms on a class of cones with fixed axis direction," *Journal of Physics A: Mathematical and General*, vol. 38, pp. 8003–8015, 2005.

[70] B. Smith, "Reconstruction methods and completeness conditions for two Compton data models," *Journal of the Optical Society of America A*, vol. 22, no. 3, pp. 445–459, 2005.

[71] V. Maxim, M. Frande, and R. Prost, "Analytical inversion of the Compton transform using the full set of available projections," *Inverse Problems*, vol. 25, no. 9, art. no. 095001, 2009.

[72] T. Hebert, R. Leahy, and M. Singh, "Three-dimensional maximum-likelihood reconstruction for an electronically collimated single-photon-emission imaging system," *Journal of the Optical Society of America A*, vol. 7, no. 7, pp. 1305–1313, 1990.

[73] R. R. Brechner and M. Singh, "Iterative reconstruction of electronically collimated SPECT images," *IEEE Transactions on Nuclear Science*, vol. 37, no. 3 Part 1, pp. 1328–1332, 1990.

[74] Y. F. Du, Z. He, G. F. Knoll, D. K. Wehe, and W. Li, "Evaluation of a Compton scattering camera using 3-D position sensitive CdZnTe detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 457, no. 1-2, pp. 203–211, 2001.

[75] A. C. Sauve, A. O. Hero III, W. L. Rogers, S. J. Wilderman, and N. H. Clinthorne, "3D image reconstruction for a Compton SPECT camera model," *IEEE Transactions on Nuclear Science*, vol. 46, no. 6, pp. 2075–2084, 1999.

[76] R. C. Rohe, M. M. Sharfi, K. A. Kecevar, J. D. Valentine, and C. Bonnerave, "The spatially-variant back-projection point kernel function of an energy-subtraction Compton scatter camera for medical imaging," in *IEEE Nuclear Science*, vol. 2, pp. 2477–2482, 1996.

[77] S. Wilderman, W. L. Rogers, G. F. Knoll, and J. C. Engdahl, "Fast algorithm for list mode back-projection of Compton scatter camera data," *IEEE Transactions on Nuclear Science*, vol. 45, no. 3, pp. 957–962, 1998.

[78] L. C. Parra, "Reconstruction of cone-beam projections from Compton scattered data," *IEEE Transactions on Nuclear Science*, vol. 47, no. 4 Part 2, pp. 1543–1550, 2000.

[79] T. Tomitani and M. Hirasawa, "Image reconstruction from limited angle Compton camera data," *Physics in Medicine and Biology*, vol. 47, pp. 2129–2145, 2002.

[80] M. Hirasawa and T. Tomitani, "An analytical image reconstruction algorithm to compensate for scattering angle broadening in Compton cameras," *Physics in medicine and biology*, vol. 48, no. 8, pp. 1009–1026, 2003.

[81] R. Basko, G. L. Zeng, and G. T. Gullberg, "Analytical reconstruction formula for one-dimensional Compton camera," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3 Part 2, pp. 1342–1346, 1997.

[82] G. T. Gullberg, G. L. Zeng, and R. Basko, "Image reconstruction from V-projections acquired by Compton camera," Nov. 24 1998. US Patent 5,841,141.

[83] I. M. Gel′fand and G. E. Shilov, *Generalized Functions, Part 1 (in Russian)*. Moscow: Gosudarstv. Izdat. Fiz.-Mat. Lit., 1958.

[84] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[85] M. Allmaras, D. Darrow, Y. Hristova, and P. Kuchment, "Detecting small low emission radiating sources," in preparation, 2010.

[86] G. I. Marchuk, *Metody Vychislitelnoi Matematiki [Methods of Numerical Mathematics]*. Moscow: Nauka, second ed., 1980.

APPENDIX A

TIME REVERSAL CODES

This appendix contains the author's Matlab implementation of the time-reversal method for thermoacoustic tomography.

There are two major parts of the code. In the first one, wave propagation is simulated and data is recorded on an *observation surface* S. In the second part wave propagation in reversed time order is performed, using the data recorded on the previous step. Below we provide more details regarding the implementation as well as the actual codes.

Step 1 The script `forward.m` is used to compute the forward propagation of waves resulting form a given initial pressure $f(x)$ (a *phantom*) and with initial velocity $g(x)$. In order to simulate wave propagation in the whole space, we solve the wave equation

$$\begin{cases} u_{tt} = c^2(x)\Delta u, \quad t \in [0, Tfinal],\ x \in D = [-a, a]^2 \\ u(x,0) = f(x), \qquad\qquad u_t(x,0) = g(x), \\ u(x,t)\mid_{\partial D} = 0, \end{cases} \tag{A.1}$$

Here the domain D is large enough to guarantee that reflections of the waves off its boundary δD do not reach the unit circle. Thus, the size of D depends on the sound speed $c(x)$ and the time $Tfinal$. The values of the wave are recorded on the boundary of the unit circle, S. The simulation of wave propagation is done using finite difference approximation, as described below. A similar implementation can be found in [26]. First, the domain D is discretized as a uniform Cartesian grid with

grid points $(x_i, y_j) = (-a + ih, -a + jh),\ i, j = 0 \ldots n,\ h = 2a/n$. The time varable $t \in [0, Tfinal]$ is discretized as $t^k = k\tau,\ k = 0 \ldots, m,\ \tau = Tfinal/m$. The solution of (A.1) at the space-time grid points, $u(x_i, y_j, t^k)$, is approximated by the (finite difference) solution $u^k_{i,j}$ of

$$
\begin{aligned}
&u^{k+1}_{i,j} = \lambda^2_{i,j}\left(u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1}\right) + (2 - 4\lambda^2_{i,j})u^k_{i,j} - u^{k-1}_{i,j}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i, j = 1, \ldots, n-1,\ k = 1, \ldots, m-1 \\
&u^0_{i,j} = f_{i,j}, \quad i, j = 0, \ldots, n \\
&u^1_{i,j} = f_{i,j} + \tau g_{i,j} + \frac{1}{2}\lambda^2_{i,j}\left(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j}\right), \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i, j = 0, \ldots, n \\
&u^k_{0,j} = u^k_{i,0} = 0, \quad i, j = 0, \ldots, n,\ k = 0, \ldots, m
\end{aligned}
\tag{A.2}
$$

Here, we have denoted $f_{i,j} := f(x_i, y_j),\ g_{i,j} := g(x_i, y_j)$ and $\lambda_{i,j} := \dfrac{c(x_i, y_j)\tau}{h}$. Equation A.2 is a second order finite difference approximation to A.1 with second order approximation of initial conditions. This numerical scheme is stable, if $\tau \leq \dfrac{h}{\sqrt{2}\max(c)}$, e.g. [86]. The observation surface S is approximated by points from the finite difference grid used for reconstruction (see Step 2) and the values of the wave at these points are obtained by nearest neighbor interpolation in space and linear interpolation in time. Care is taken so that the meshes for forward wave propagation and reconstruction are different. The data collected on the observation surface is saved in the file `forward_data.m`.

Step 2 The time-reversal method is implemented in the script `reconstruction.m`. The goal is to solve the wave equation inside the unit circle, starting at time $T \leq$

$Tfinal$ until time 0. The initial conditions at $t = T$ are set to zero and the boundary conditions, which were recorded at the previous step, are transmitted to S in reversed time order. In the actual implementation we solve equation (A.2) inside the square $[-1.2, 1.2]^2$, from time $t = T$ until time $t = 0$, with zero initial conditions at $t = T$ and zero boundary conditions. The values of the wave on S are enforced on the solution on each time step. As a result, we obtain the correct solution of the wave equation inside the unit disc, which is our area of intereset.

Below is the order in which the codes should be run:

1. In order to set up the parameters, modify and run configure.m

2. For simulating wave propagation, run forward.m.

3. Once the data on the observation surface is recorded, one can experiment with reconstructions from different cut-off times T. In order to reconstruct the phantom, run reconstruction.m. This last script will prompt for a value of the cut-off time T.

Listing IV.1 'configure.m'

```
% Forward problem: Solve the wave equation
% u_tt = c^2\Delta u in the domain
% D = [-a,a]^2 (a>2) with initial conditions
% u=phantom, % du/dt=initial_velocity and
% 0 values on the boundary. The values of
% the solution are recorded on the boundary of
% the unit circle.
```

```
clear all

% reqire: c*Tfinal/2 + 1 < a
%-------------------------------------------------------------%
%          Parameters for the Forward Wave                    %
%-------------------------------------------------------------%
                              % Compute the wave in [-a,a]^2
a = 3;
                              % h = 2*a/n will be the mesh size
n = 800;
                              % The time till which the solution
                              % will be computed.
Tfinal = 3.05;

%------------ Soundspeed for the forward wave ----------------
cd Speeds
c = nice_cos(a,n);
cd ..

%------------ Set Initial Conditions -------------------------
cd Phantoms
                              % initial pressure
phantom = comb1_phantom(a,n);
                              % initial velocity
initial_velocity = zeros(n+1,n+1);
cd ..

%-------------------------------------------------------------%
%                  Reconstruction Parameters                  %
%-------------------------------------------------------------%
                              % Compute the reconstruction
                              % on [-ar,ar]^2
ar=1.2;
                              % hr = 2*ar/nr will be the mesh size
nr = 400;
                              % The following is done to ensure
                                 the
                              % grids for the forward wave and for
                              % the reconstruction are different
if 2*ar/nr == 2*a/n
    nr = nr-10;
end

%--------- Soundspeed for the reconstruction ---------------
cd Speeds
```

```
cr = nice_cos(ar,nr);
cd ..

%———————— Limited View ——————————————————————————
                              % Data is NOT available for parts of
                              % the boundary with polar
                                 coordinates
                              % in (ang1,ang2). 'ang1' and 'ang2'
                              % must be given in radians and
                              % 0 <= ang1 < ang2 <= 2*pi.
ang1 = 0; ang2 = pi/4;

%—————Create the phantom on the reconstruction mesh——————
%          Used to compare with the reconstruction
cd Phantoms
phantomR = comb1_phantom(ar,nr);
cd ..

save config
disp('Configuration done.')
```

Listing IV.2 'forward.m'

```
clear all, clf reset
disp('forward starts....')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Setup                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                              % Load parameters for the forward
                              % solution, including sound speed
                              % and initial conditions.
load config

x=linspace(-a,a,n+1); [X,Y]=meshgrid(x);
h = 2*a/n;
                              % Compute the approximation of the
                              % circle Br with points from the
                              % reconstruction grid (UbX, UbY), at
                              % which the values of the wave will
                              % be recorded.
xr=linspace(-ar,ar,nr+1);
[UbX,UbY,indexX,indexY] = find_boundary(1,xr);
```

```
                        % Plot initial conditions
ang = 0:.01:2*pi;
gray_plot(X,Y,phantom)
title('Initial Pressure')
hold
plot3(UbX,UbY,zeros(length(UbX)),':','LineWidth',2)
hold off
pause

gray_plot(X,Y,initial_velocity)
title('Initial Velocity')
hold
plot3(UbX,UbY,zeros(length(UbX)),':','LineWidth',2)
hold off
pause

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Finite Difference Solution                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Parameters
%-------------------------------------------------------------------------
                        % time step satisfying the stability
                        % cond.for the forward mesh
tauf = h/(1.42*max(c(:)))
time_steps = ceil(Tfinal/tauf), tauf = Tfinal/time_steps;
fwdtimes = linspace(0,Tfinal,time_steps+1);

% Begin Algorithm
%-------------------------------------------------------------------------
t=0; timestep = 1;
L=(c.*tauf/h).^2;
Uoldest = phantom;
                        % Record values at the circle Br
Ubound(:,timestep)= interp2(X,Y,Uoldest,UbX,UbY,'*nearest');


                        % Set the values for U at time "tau"
                        % using 2nd order approx. of
                        % initial velocity
t = t+tauf; timestep = timestep + 1;
Uold = Uoldest + tauf*initial_velocity + (L./2).*...
   (circshift(Uoldest,[1 0])+circshift(Uoldest,[-1 0])+...
    circshift(Uoldest,[0 1])+circshift(Uoldest,[0 -1]) -...
```

```
                                                    4*Uoldest);
                              % Set boundary of [-a,a]^2 to 0
Uold(1,:)=0; Uold(n+1,:)=0;
Uold(:,1)=0; Uold(:,n+1)=0;

Ubound(:,timestep) = interp2(X,Y,Uold,UbX,UbY,'*nearest');

while t < Tfinal
  t = t+tauf, timestep = timestep + 1;
  Utemp=L.*(circshift(Uold,[1 0])+circshift(Uold,[-1 0])+...
            circshift(Uold,[0 1])+circshift(Uold,[0 -1]))+...
                                          (2-4*L).*Uold - Uoldest;
  Uoldest = Uold;
  Uold = Utemp;
  Uold(1,:)=0; Uold(n+1,:)=0;
  Uold(:,1)=0; Uold(:,n+1)=0;

  Ubound(:,timestep) = interp2(X,Y,Uold,UbX,UbY,'*nearest');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Graphs                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gray_plot(X,Y,Uold)
title('Solution at time Tfinal')
hold
plot3(UbX,UbY,zeros(length(UbX)),':','LineWidth',2)
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Save Data                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save forward_data Ubound UbX UbY indexX indexY fwdtimes
Tfinal tauf h

disp('E N D (forward)')
```

Listing IV.3 'reconstruction.m'

```
% Reconstruct the wave inside the circle with radius 'r'

clear all, clf reset
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                    Input Parameters                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load config
                              % Load data Ubound UbX
                              % UbY indexX indexY
                              % fwdtimes Tfinal tauf h
load forward_data

T = Tfinal;
while (T >= Tfinal)
    fprintf('Enter time from which to reconstruct'
        '(must be less than %3.2f): \n',Tfinal );
    T = input('T = ');
end

Ubound = limited_view(Ubound,ang1,ang2);

                              % Reconstruction mesh
xr=linspace(-ar,ar,nr+1); [Xr,Yr]=meshgrid(xr);
hr = 2*ar/nr;
                              % 'taur' - reconstruction time step
taur = hr/(1.42*max(c(:)));
time_steps = ceil(T/taur); taur = T/time_steps;

bkwdtimes = linspace(0,T,time_steps+1);
tauf, taur
                             % Interpolate boundary data
UB = interpolate_boundary(Ubound,fwdtimes,bkwdtimes);
                             % Flip the time
UB(:,1:end) = UB(:,end:-1:1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Begin Algorithm                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t = T, timestep = 1,

% Set initial conditions:  U at times 0 and tau equals 0
Uoldest = zeros(nr+1,nr+1);

t = T-taur, timestep = timestep + 1;
Uold = Uoldest;

                               % enforce boundary conditions
for i=1:length(indexX)
```

```
    Uold(indexY(i),indexX(i)) = UB(i,timestep);
end

L=(cr.*taur/hr).^2;

while t >= taur

   t=t-taur,  timestep = timestep+1;
   Utemp = L.*(  circshift(Uold,[1  0])  +...
          circshift(Uold,[-1  0]) + circshift(Uold,[0  1])+...
          circshift(Uold,[0  -1])  ) + (2-4*L).*Uold - Uoldest;
   Uoldest = Uold;
   Uold = Utemp;

                              % enforce boundary conditions
   for i=1:length(indexX)
        Uold(indexY(i),indexX(i)) = UB(i,timestep);
   end

end % while

                              % Initial velocity reconstruction
velocity_rec = (Uoldest-Uold)/taur- L/(2*taur).*...
   ( circshift(Uold,[1  0]) + circshift(Uold,[-1  0])  +...
     circshift(Uold,[0  1]) + circshift(Uold,[0  -1])  -...
                                                  4*Uold  );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           Errors                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                              % cut away the values outside the
                              % unit circle
Uold_cut = Uold.*double(Xr.^2+Yr.^2<0.97);
Phantom_cut = phantomR.*double(Xr.^2+Yr.^2<0.97);
error = Uold_cut-Phantom_cut;
[dedx,dedy] = gradient(error,h);
[dPhantom_dx,dPhantom_dy]=gradient(Phantom_cut,h);

l2errP=100*sqrt(sum(sum(error.^2))/sum(sum(Phantom_cut.^2)))
h1errP = 100*sqrt(sum(sum(error.^2+dedx.^2+dedy.^2))/...
    sum(sum(Phantom_cut.^2+dPhantom_dx.^2+dPhantom_dy.^2 )))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           Graphs                          %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                  % Call the scropt mygraphs.m
mygraphs
```

Below are the auxiliary functions and scripts used in the main scripts `forward.m` and `reconstruction.m`.

Listing IV.4 'find_boundary.m'

```
function [UbX,UbY,indexX,indexY] = find_boundary(r,x)

% Approximates the boundary of the circle centered at 0 with
% radius 'r' by points from the square mesh
% [X,Y] = meshgrid(x).
%
% A point from the mesh is a boundary point if at least one
% of its 4 neighbours is outside the circle.
%
% Output:
% ----------
% UbX, UbY hold the coordinates of the boundary points. The
% vectors 'indexX' and 'indexY' hold the indexes
% (corresponding to the mesh X,Y) of the boundary points.

[X,Y] = meshgrid(x);
UbX = [];
UbY = [];
indexX =[];
indexY =[];

h = X(1,2)-X(1,1);                              % mesh size

for i = 1:max(size(X))
for j=1:max(size(Y))
           norm_ij = norm([X(i,i),Y(j,j)]);
           if(norm_ij<=r&&norm_ij>r-h*sqrt(2))
           norm_mo = norm([X(i-1,i-1),Y(j,j)]);
           norm_po = norm([X(i+1,i+1),Y(j,j)]);
           norm_om = norm([X(i,i),Y(j-1,j-1)]);
           norm_op = norm([X(i,i),Y(j+1,j+1)]);
           if(norm_mo>r || norm_po>r || norm_om>r || norm_op>r)
                    UbX=[UbX, X(i,i)];
```

```
                UbY=[UbY, Y(j,j)];
                indexX =[indexX, i];
                indexY =[indexY, j];
        end
        end
end
end

%-------------Sort UbX, UbY, indexX and indexY-----------------%
%  by increasing angle coordinate of the boundary points
%                [UbX(i),UbY(i)]
%--------------------------------------------------------------%
                        % get the angular coordinates of the
                        % boundary points.
[phi,rho] = cart2pol(UbX,UbY);
                        % The angular coordinates "phi" are
                        % then sorted by increasing angle,
                        % "order" is the vector of indeces
                        % such that phi_ordered=phi(order).
[phi_ordered,order] = sort(phi);
clear phi rho phi_ordered

                        % Now reorder UbX, UbY, indexX and
                        % indexY.
UbX = UbX(order);
UbY = UbY(order);
indexX = indexX(order);
indexY = indexY(order);
```

Listing IV.5 'limited_view.m'

```
function [LVUbound] = limited_view(Ubound,ang1,ang2)
% Produces a limited view data by (smoothly) replacing the
% boundary data corresponding to points on the boundary with
% polar coordinates in (ang1, ang2) by zero. 'ang1' and
% 'ang2' must be given in radians and
% 0 <= ang1 < ang2 <= 2*pi.

alpha = ang1;
beta= ang2;

if (alpha - beta) ==0
    LVUbound = Ubound;
```

```
else

                        % h_bound is approximately the
                        % discretization step along the
                        % boundary of the unit circle.
    h_bound = 2*pi/(length(Ubound(:,1))-1);

                        % alpha_i, beta_i are the indexes in
                        % Ubound that correspond to the
                        % angles alpha and beta
    alpha_i = floor(alpha/h_bound)+1;
    beta_i = ceil(beta/h_bound)+1;

                        % we will smoothly cut to zero
                        % within 20% from either side.
    smooth = floor((beta_i-alpha_i)/5);

    LVUbound = Ubound;
    LVUbound(alpha_i+1:beta_i-1,:)= 0;
    for j=1:length(Ubound(1,:))
      LVUbound(alpha_i:alpha_i+smooth,j)=...
      LVUbound(alpha_i,j).*(1+ cos(pi*(0:smooth)/smooth))/2;
      LVUbound(beta_i-smooth:beta_i,j)=...
      LVUbound(beta_i,j).*(1- cos(pi*(0:smooth)/smooth))/2;
    end
end %if/else
```

Listing IV.6 'interpolate_boundary.m'

```
function [UboundR] = interpolate_boundary(Ubound,fwdtimes,
                                                  bkwdtimes)
% Linear interpolation of Ubound given at fwdtimes to
% bkwrdtimes. Ubound(:,i) is the vector containing the
% values for time fwdtimes(i).

if fwdtimes(end)<bkwdtimes(end)
    error('T must be less than to Tfinal')
    return
end

tauf = abs(fwdtimes(1)- fwdtimes(2));
taur = abs(bkwdtimes(1)- bkwdtimes(2));
```

```
Nf = length(fwdtimes); Nr = length(bkwdtimes);

for i = 1:Nr
    p = (i-1)*taur;
    k = floor(p/tauf)+1;
    a = (Ubound(:,k+1)-Ubound(:,k))/tauf;
    b = (1-k)*Ubound(:,k+1)+k*Ubound(:,k);
    UboundR(:,i) = a*p+b;
end
```

The next three functions are samples of a phantom and two sound speeds.

Listing IV.7 'comb1_phantom.m'

```
function [U] = comb1_phantom(a,n)
% Produces the characteristic function of the part of the
% annulus {a<=|X|<=b} which lies in the first quadrant +
% two circles.
% The output 'U' gives the values of the phantom on the
% square grid on [-a,a]^2 with mesh size 2a/n.

[X,Y] = meshgrid(linspace(-a,a,n+1));
a=0; b = 0.7;
U=double((X.^2+Y.^2>=a^2)).*((X.^2+Y.^2<=b^2)).*(X>=0).*...
                (Y>=0)+0.7*double( (X+0.5).^2+Y.^2<=0.09 )+...
                                1.3*double(X.^2+(Y+0.5).^2<=0.01);
```

Listing IV.8 'nice_cos.m'

```
function [c] = nice_cos(a,n)
% Time of travel though the unit circle: ~ 1.694

[X,Y] = meshgrid(linspace(-a,a,n+1));
c = (1+0.2*cos(X.^2+Y.^2)).*double(X.^2+Y.^2<=1) +...
                            (1+0.2*cos(1)).*double(X.^2+Y.^2>1);
```

Listing IV.9 'parabolic_speed.m'

```
function [c] = parabolic_speed(a,n)
```

```
% c = 0.1 +X^2+Y^2
%Time it takes a wave to cross the unit circle: ~8

[X,Y] = meshgrid(linspace(-a,a,n+1));
c=0.1+(X.^2+Y.^2).*double(X.^2+Y.^2<=1)+double(X.^2+Y.^2>1);
```

The following script was used for numerical estimation of the error as a function of cut-off time T (see 2.5).

Listing IV.10 'reconstruction_loop.m'

```
% This script was written to compute the dependence of the
% approximation error on the cut-of time T. In order to run
% this, comment the part in reconstruction.m which defines T
% as well as the "clear all" command
% (both are at the very beginning of reconstruction.m)

clear all, clf reset

load forward_data Tfinal
% The reconstructions are computed for T in [t0, Tfinal-0.05]
t0 = 2;
reconstructionT = [t0:0.2:Tfinal-0.05];

E = [];
for recT = 1:length(reconstructionT)
    T = reconstructionT(recT)
    reconstruction
    E(recT,:) = [reconstructionT(recT) l2errP h1errP];
end

save err_wrt_T E

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                   Graphs                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(E(:,1),E(:,2),'-k','LineWidth',2)
hold on
plot(E(:,1),t0*E(1,2)./E(:,1),':k','LineWidth',2)
legend('L^2 error ','C/T')
%title('Trapping sound speed')
xlabel('cut-off time T')
```

```
ylabel('L^2 error')
hold off
pause
plot(E(:,1),E(:,3),'-k','LineWidth',2)
hold on
plot(E(:,1),t0*E(1,3)./E(:,1),':k','LineWidth',2)
legend('H^1 error ','C/T')
%title('Trapping sound speed')
xlabel('cut-off time T')
ylabel('H^1 error')
hold off
pause

%       Logarithmic Scale and Linear Regression
%-------------------------------------------------------------------
%
logE=log(E);

pl2_str=polyfit(logE(:,1),logE(:,2),1);
pl2=polyval(pl2_str,logE(:,1));
fprintf('Polyfit for L^2 error is: %6.2f x+%6.2f\n',pl2_str)
ph1_str=polyfit(logE(:,1),logE(:,3),1);
ph1=polyval(ph1_str,logE(:,1));
fprintf('Polyfit for H^1 error is: %6.2f x+%6.2f\n',ph1_str)

plot(logE(:,1),logE(:,2),'-k','LineWidth',2)
hold on
plot(logE(:,1),pl2,':k','LineWidth',2)
legend('log of L^2 error ',[num2str(pl2_str(1)),'x+',
                                        num2str(pl2_str(2))])
xlabel('log T')
ylabel('log of L^2 error')
hold off
pause

plot(logE(:,1),logE(:,3),'-k','LineWidth',2)
hold on
plot(logE(:,1),ph1,':k','LineWidth',2)
legend('log of H^1 error ',[num2str(ph1_str(1)),'x+',
                                        num2str(ph1_str(2))])
xlabel('log T')
ylabel('log of H^1 error')
hold off
pause
```

```
%        Scaled graphs for trapping speeds *only*
%-------------------------------------------------------------------
%
time = log(log(E(:,1)+2));

logE=log(E);

pl2_str=polyfit(time,logE(:,2),1);
pl2=polyval(pl2_str,time);
fprintf('Polyfit for L^2 error is: %6.2f x+%6.2f\n',pl2_str)
ph1_str=polyfit(time,logE(:,3),1);
ph1=polyval(ph1_str,time);
fprintf('Polyfit for H^1 error is: %6.2f x+%6.2f\n',ph1_str)

plot(time,logE(:,2),'-k','LineWidth',2)
hold on
plot(time,pl2,':k','LineWidth',2)
legend('log of L^2 error ',[num2str(pl2_str(1)),'x+',
                                              num2str(pl2_str(2))])
xlabel('log log (2+T)')
ylabel('log of L^2 error')
hold off
pause

plot(logE(:,1),logE(:,3),'-k','LineWidth',2)
hold on
plot(logE(:,1),ph1,':k','LineWidth',2)
legend('log of H^1 error ',[num2str(ph1_str(1)),'x+',
                                              num2str(ph1_str(2))])
xlabel('log T')
ylabel('log of H^1 error')
hold off
```

The next three files are used for the graphs.

Listing IV.11 'mygraphs.m'

```
ang = ang1:-.01:-2*pi+ang2;
                                                  % Grayscale
figure(1)
subplot(1,2,1);
gray_plot(Xr,Yr,Phantom_cut)
```

```
title('Phantom')
hold
plot3(cos(ang),sin(ang),ones(1,length(ang)),':w',
                                              'LineWidth',2)
hold off
subplot(1,2,2)
gray_plot(Xr,Yr,Uold_cut)
title(['Reconstruction from time T=',num2str(T),'  L2 err ',
       num2str(l2errP),' %, H1 err ',num2str(h1errP),' %'])
hold
plot3(cos(ang),sin(ang),ones(1,length(ang)),':w',
                                              'LineWidth',2)
hold off
                                              % Surface
figure(2)
subplot(1,2,1);
nice_plot(Xr,Yr,Phantom_cut)
title('Phantom')
hold
plot3(cos(ang),sin(ang),zeros(1,length(ang)),':',
                                              'LineWidth',2)
hold off
subplot(1,2,2)
nice_plot(Xr,Yr,Uold_cut)
title(['Reconstruction from time T=',num2str(T),'  L2 err ',
       num2str(l2errP),' %, H1 err ',num2str(h1errP),' %'])
hold
plot3(cos(ang),sin(ang),zeros(1,length(ang)),':',
                                              'LineWidth',2)
hold off

                          % Profile plots at y= y0

y0 = 0;
                          % 'at' determines the entries of
                          % the arrays corresponding to 'y0'
at =floor((ar+y0)/hr);
figure(3)
subplot(1,2,1)
gray_plot(Xr,Yr,Uold_cut)
hold on
plot3(xr,y0*ones(size(xr)),max(max(Uold_cut))*
                            ones(size(xr)),'-','LineWidth',1)
hold off
```

```
title('Reconstruction')
subplot(1,2,2)
plot(xr,Uold_cut(at,:),'-k','LineWidth',2)
axis([-1 1 -0.1, 1.2])
hold on
plot(xr,Phantom_cut(at,:),':k','LineWidth',2)
xlabel('x')
ylabel('y')
legend(['Reconstruction form T = ',num2str(T)],'Phantom')
title(['y=',num2str(y0),' profile;  l2 Error ',
        num2str(l2errP),' %, h1 err ',num2str(h1errP),' %'])
hold off
```

Listing IV.12 'gray_plot.m'

```
function [] = gray_plot(X,Y,U)
Uplot = U;
surf(X,Y,Uplot)
colorbar
xlabel('x')
ylabel('y')
colormap(gray)
view(0,90)
shading interp
```

Listing IV.13 'nice_plot.m'

```
function [] = nice_plot(X,Y,U)
Uplot = U;
surf(X,Y,Uplot)
colorbar
xlabel('x')
ylabel('y')
zlabel('u')
colormap(jet)
view(7,43)
shading interp
lightangle(-45,30)
set(gcf,'Renderer','zbuffer')
set(findobj(gca,'type','surface'),...
'FaceLighting','phong',...
```

```
'AmbientStrength',.3,'DiffuseStrength',.8,...
'SpecularStrength',.9,'SpecularExponent',25,...
'BackFaceLighting','unlit')
```

APPENDIX B

COMPTON CAMERA IMAGING CODES

This appendix contains the author's Matlab codes for inversion of 2D cone transform and for simulation and detection of a geometrically small source in the presence of a large background. Numerical examples based of these codes are presented in Chapter III.

First, analytical cone projections of phantoms were computed. Four detector arrays were placed on the sides of the unit square, i.e. the vertices of the cones were restricted to the sides of the square. Methods A,B and C were used to convert cone projections into Radon projections and then filtered backprojection (1.15) was utilized to invert the Radon transform and obtain reconstructions of the phantoms. Examples of reconstructions using data from one and three detector arays can be found in Section 3.3.

We also simulated Compton and Radon emission data from point sources in the presence of a large random background radiation. The same reconstruction methods as for analytic data were used for detection of sources. The results of these computations were not entirely satisfactory. Reconstructions and a discussion were presented in Section 3.5.1. Next, backprojection was utilized to detect sources. This approach offers several advantages over standard tomography reconstructions, as described in Section 3.4.2. In particular, it allows for treatment of non-uniform background. Reconstructions and a detailed discussion of the numerical results can be found in Section 3.5.2.

We proceed with a brief description of the codes and instructions for their use.

The set up of parameters is done in the script `configure.m` (Listing IV.14). Here, the location and the centers of bins of detector arrays are specified, as well as the discretization of cones' central axes and scattering angles. The configuration is saved in the file `config.mat`.

The script `makephantom.m` (Listing IV.15) creates phantom data based on the saved configuration. Radon and cone projection data of phantoms consisting of several circular inclusions can be created by choosing the option for analytic phantom. If the source phantom option is chosen, the code simulates Radon and Compton-type detector data coming from several point sources and a random background. The data is saved in the file `phantomdata.mat`.

There are three codes which can be used for reconstruction. The script `cone_inversionA.m` (Listing IV.16) is an implementation of method A for inversion of the cone transform. The script `reconstruct.m` (Listing IV.17) has an option for using methods B or C for transforming cone into fanbeam data. Then filtered backprojection is used to invert the Radon transform. The backprojection method for detection of geometrically small sources is implemented in `reconstruct_pointsource.m` (Listing IV.18). The last script should be used for emission-type data only, while the first two could be used for projection data as well as for emission type data. Both `reconstruct.m` and `reconstruct_pointsource.m` have an option for the number of detector arrays used for reconstruction. These codes can used for reconstructions from Radon or Compton type data.

The programs should be run in the following order:

1. In order to set up the parameters, modify and run `configure.m`

2. To create a phantom, run `makephantom.m`. This script prompts for a type of phantom (analytic projections or emission-type) and for its specifics.

3. Once the phantom data is computed, the phantom can be reconstructed by `cone_inversionA.m`, `reconstruct.m` or `reconstruct_pointsource.m`. Different number of detector arrays can be used.

Listing IV.14 'configure.m'

```
clear all
                    % #bins in each detector array +1
ndet = 100;
                    %# rays from each detector (alpha)
nalpha = 200;
                    % # central axes beta
nbeta = nalpha-2;
                    % # scattering angles psi
npsi = nbeta;
                    % alpha -- rays from each detector, i.e.
                    %  angular coordinates
alpha = linspace(0,pi,nalpha);% alpha = alpha(2:nalpha+1);
                    % phi=alpha-pi/2
phi = linspace(-pi/2,pi/2,nalpha);% phi = phi(2:nalpha+1);
                    % half-angles psi
psi = linspace(0,pi,npsi+2); psi = psi(2:npsi+1);
                    % central axes beta
beta = linspace(0,pi,nbeta+2); beta = beta(2:nbeta+1);

%-------------------------------------------------------------%
%                        Detectors                            %
%-------------------------------------------------------------%
disp('Detector arrays are placed on the sides of the unit
   square.')
%
%
disp('                              detector                 ')
disp('                              array III                ')
```

```
disp('                          ____________________              ')
disp('           detector      |                    | detector    ')
disp('           array IV      |                    | array II    ')
disp('                         |         *          |             ')
disp('                         |                    |             ')
disp('                          ____________________              ')
disp('                       -1     detector     1                ')
disp('                              array I                       ')

           % Coordinates of the detectors.
x1(:,1) = linspace(-1,1,ndet);
x2(:,1) = -1*ones(size(x1(:,1)));

x1(:,3) = linspace(-1+1/(7*ndet),1+1/(7*ndet),ndet);
x2(:,3) = 0.999*ones(size(x1(:,2)));

x2(:,2) = linspace(-1-1/(11*ndet),1-1/(11*ndet),ndet);
x1(:,2) = ones(size(x2(:,3)));

x2(:,4) = linspace(-1+1/(13*ndet),1-1/(13*ndet),ndet);
x1(:,4) = -1*ones(size(x2(:,4)));

save config
disp('Configuration done!')
```

Listing IV.15 'makephantom.m'

```
clear all
load config
phan = ' ';
while (phan ~='a' & phan~='s')
    fprintf('Phantom: a = analytic, s = source\n');
    phan = input('Enter phantom type = ','s');
end

if phan == 'a'
    disp('Analytic phantom consisting of circular inclusions
        ...')
    ndisks = input('Enter number of inclusions = ');
    for i=1:ndisks
        c(:,i) = input('Center of disk [x,y]=')';
        r(i) = input('radius = ');
        a(i) = input('amplitude =');
```

```
    end
    [C U] = ...
    analyticphantom(c(1,:),c(2,:),r,a,alpha,psi,beta,x1,x2);
    save phantomdata C U c r a x1 x2 alpha psi beta
else
    disp('Source phantom....')
    nsources = input('Enter number of sources = ');
    for i=1:nsources
        S(:,i) = input('Location of point source inside unit
            square [x,y]=')';
    end
    bkgd = input('Number of background particles = ');
    ratio = input('SNR (ratio of # particles from one source
        to # background particles)=')
    [C, U, sourcepart, bkgdpart] = ...
        sourcephantom(S,bkgd,ratio,alpha,psi,beta,x1,x2);
    save phantomdata C U S sourcepart bkgdpart x1 x2...
                                                alpha psi beta
end
disp('Phantom data ready!')
mybeep
```

Listing IV.16 'cone_inversionA.m - Inversion of the cone transform via method A'

```
% Inversion of 2D cone transform via method A. Here the
% vertices of cones (i.e. detectors) are
% restricted to a horizontal line. The phantom must be
% supported above the detector array.

clear all
%=========================================================%
%                      Setup                              %
%=========================================================%
load phantomdata
ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);

%=========================================================%
%          Integration with respect to psi                %
%=========================================================%
for k=1:npsi
Gd(:,:,k) = ((psi(k)-pi/2)./sin(psi(k)-pi/2)).*C(:,:,k,1);
end
```

```
                        % Take Hilbert transform
for   i = 1:ndet
    for j = 1:nbeta
    HGd(i,j,:)=hilbert(squeeze(Gd(i,j,:)));
    end
end
HGd = imag(HGd);
                        % Take the value at 0
ICd=-squeeze(HGd(:,:,npsi/2));

%=========================================================%
%       Interpolate to parallel beam geometry             %
%=========================================================%
sd = x1(:,1)*cos(beta)+x2(:,1)*sin(beta);
nlines=4*ndet;
s=linspace(-sqrt(2),sqrt(2),nlines);

for j=1:nbeta
    HFd(:,j)=interp1(sd(:,j),ICd(:,j),s,'linear');
    nan = isnan(HFd(:,j))';
    [m1,at1] = min(nan); [m2,at2] = min(fliplr(nan));
    at2 = length(HFd(:,j))-at2+1;
    HFd(1:at1-1,j)=HFd(at1,j); HFd(at2+1:end,j)=HFd(at2,j);
    clear nan m1 m2 at1 at2
end
%=========================================================%
%            Inverse Hilbert                              %
%=========================================================%
for j=1:nbeta
    Gd(:,j)=-hilbert(HFd(:,j));
end
                    % 'Gd' is the Radon transform of the phantom
Gd = imag(Gd);

%=========================================================%
%                    Radon invesrion                      %
%=========================================================%
res = ndet+20; % resolution of the output image
xb = linspace(-1.1,1.1,res);  yb=xb;
                            % domain of the output image
[XB,YB]=meshgrid(yb,xb);
                            % Invert using FBP
Rec = radon_inversion(Gd,s,beta,xb,yb,1);
%=========================================================%
```

```
%                          Graphs                                    %
%====================================================================%
gray_plot(XB,YB,Rec);
view(0,90)
hold on
plot3(x1(:,1),x2(:,1)-0.01,0*ones(1,length(x1(:,1))),...
                                                    ':g','LineWidth',2)
hold off
disp('Reconstruction done!')
```

Listing IV.17 'reconstruct.m - Inversion of the cone transform via methods B and C'

```
% Filtered backprojection reconstruction of 2D Radon or
% Compton data, collected by detector arrays placed on sides
% of the unit square. Radon data, which is given in fanbeam
% coordinates, is interpolated to parallel beam data.
% Then filtered backprojection is used for reconstructing
% the phantom. Compton data is converted to
% fanbeam data by the use of methods B or C
% (cone2fanbeamB.m or cone2fanbeamC.m),
% then the procedures described above are applied.
%
%                              detector
%                              array III
%                          ---------------
%              detector    |             |  detector
%              array IV    |      *      |  array II
%                          |             |
%                          |             |
%                          ---------------
%                              detector
%                              array I
clear all
%====================================================================%
%                        Setup                                       %
%====================================================================%
phan = ' ';
while (phan ~='a' & phan~='s')
    fprintf('Phantom: a = analytic, s = source\n')
    phan = input('Enter type =','s');
end

data = ' ';
```

```
while (data ~= ['r','c'])
    fprintf('Type of data: r - Radon, c - Cone \n')
    data = input('data = ','s');
end

method = ' ';
if data == 'c'
   while (method~=['B','C'])
   fprintf(...
   'Choose method for reduction to fanbeam data - B or C\n')
   method = input('method = ','s');
   end
end

numdet = 0;
while (numdet~=[1 2 3 4])
    numdet = input('Number detectors (1--4): ');
end
                          % Filters for inversion of Radon
                          % transform. filter can be a vector
                          % containing integers from 1 to 8.
                          % Filters are described in
                          % radon_inversion.m
filter = [1];
                           % Load data from file
load phantomdata
ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);


%=========================================================%
%            Convert Cone to Fanbeam                      %
%=========================================================%
if data == 'c'
   clear U
   if method == 'B'
      for d=1:numdet
       U(:,:,d) = cone2fanbeamB(C(:,:,:,d));
      end
   else
        for d=1:numdet
           U(:,:,d) = cone2fanbeamC(C(:,:,:,d),phan);
        end
   end
end
```

```
%=========================================================%
%      Convert Fanbeam to Parallel Beam                   %
%=========================================================%
                              % Fanbeam coordinates of a line are
                              % a point (detector) (x1,x2) and
                              % an angle "alpha" between the line
                              % and the positive x1-axis.
                              % Radon coordinates are (s, phi),
                              % where "phi" is the angular
                              % coordinate of the vector
                              % perpendicular to the line and "s"
                              % is the distance of the line from
                              % the origin.
nlines=4*ndet;
s=linspace(-sqrt(2),sqrt(2),nlines);
                              % sfan(:,:,d) are the available
                              % "s"-values for the corresponding
                              % detectors. For each fixed value of
                              % phi (i.e. phi(j)) sfan(:,j,d) is a
                              % column vecor containing the values
                              % of "s" for the angle phi(j) and
                              % detector "d".
phi = alpha-pi/2;
for d=1:numdet
    sfan(:,:,d) = x1(:,d)*cos(phi)+x2(:,d)*sin(phi);
end

spar=[]; Upar = [];
for d=1:numdet
    spar = [spar; sfan(:,:,d)];
    Upar = [Upar; U(:,:,d)];
end

G(:,1)=zeros(nlines,1); G(:,length(phi))=zeros(nlines,1);
for j=2:length(phi)-1
    G(:,j)=interp1(spar(:,j),Upar(:,j),s,'linear');
                              % Extrapolate by constant.
                              % at1 and at2 are the first and last
                              % index of G(:,j) at which the values
                              % were interpolated.
    nan = isnan(G(:,j))';
    [m1,at1] = min(nan); [m2,at2] = min(fliplr(nan));
    at2 = length(G(:,j))-at2+1;
```

```
    G(1:at1-1,j)=G(at1,j); G(at2+1:end,j)=G(at2,j);
end

%=========================================================%
%        Reconstruct Image using Radon Inversion          %
%=========================================================%
ndim = ndet;
xb = linspace(-0.98,0.98,ndim);
yb = linspace(-0.98,0.98,ndim);
[XB,YB]=meshgrid(xb,yb);
for f = filter
   FBP(:,:,f) = radon_inversion(G(:,:),s,phi,xb,yb,f);
end

%=========================================================%
%                        Error                            %
%=========================================================%
if (phan == 'a')
    exact = zeros(size(XB));
    for i = 1:length(c(1,:))
      exact = exact+a(i)*ball(c(1,i),c(2,i),r(i),XB,YB);
    end
    exact_cut = exact((abs(XB)<0.98)&(abs(YB)<0.98));
    approx = squeeze(FBP(:,:,1));
    approx_cut = approx((abs(XB)<0.98)&(abs(YB)<0.98));
    error = exact_cut - approx_cut;
    l2errP=100*sqrt(error'*error)/sqrt(exact_cut'*exact_cut)
end

%---------------------------------------------------------
graphrec
disp('Compton reconstruction done !!!')
```

Listing IV.18 'reconstruct_pointsource.m - Reconstruction of source distribution via backprojection'

```
% Backprojection of 2D source data (Radon or Compton),
% collected by detector arrays placed on sides of the unit
% square. Radon data, which is given in fanbeam
% coordinates, is interpolated to parallel beam data. Then,
% the data from each detector array is backprojected
% separately, after which the backprojections are added
% Local grids surrounding each pixel from the backprojected
```

```
% image are used to estimate the (local) mean and (local)
% standard deviation. A threshold value for the number
% (local) standard deviations above the (local) mean is used
% for estimating the confidence of source detection.
% Compton data is converted to Radon data via
% method C (cone2fanbeamC.m), then the procedures described
% above are applied.
% The geometry of the detectors is as follows
%
%                              detector
%                              array III
%                           ---------------
%              detector     |             |  detector
%              array IV     |      *      |  array II
%                           |             |
%                           |             |
%                           ---------------
%                              detector
%                              array I
clear all
%=========================================================%
%                        Setup                            %
%=========================================================%
data = ' '; numdet = 0;
while (data ~= ['r','c'])
    fprintf('Type of data: r - Radon, c - Cone \n');
    data = input('data = ','s');
end
while (numdet~=[1 2 3 4])
    numdet = input('Number detectors (1--4): ');
end
                              % Load data from file
load phantomdata
ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);

%=========================================================%
%              Convert Compton to Fanbeam                 %
%=========================================================%
if data == 'c'
   clear U
   for d=1:numdet
       U(:,:,d) = cone2fanbeamC(C(:,:,:,d),1);
   end
```

```
end

%=========================================================%
%      Convert Fanbeam to Parallel Beam                   %
%=========================================================%
                              % Fanbeam coordinates of a line are
                              % a point (detector) (x1,x2) and
                              % an angle "alpha" between the line
                              % and the positive x1-axis.
                              % Radon coordinates are (s, phi),
                              % where "phi" is the angular
                              % coordinate of the vector
                              % perpendicular to the line and "s"
                              % is the distance of the line from
                              % the origin.
nlines=4*ndet;
s=linspace(-sqrt(2),sqrt(2),nlines);

                              % sfan are the available
                              % "s"-values for the corresponding
                              % detectors. For each fixed value of
                              % phi (i.e. phi(j)) sfan(:,j,d) is a
                              % column vector containing the
                                values
                              % of "s" for the angle phi(j) and
                              % detector "d"
phi = alpha-pi/2;
for d=1:numdet
sfan(:,:,d) = x1(:,d)*cos(phi)+x2(:,d)*sin(phi);
end

for j=1:nalpha % phi = alpha-pi/2
    for d=1:numdet
        G(:,j,d)=interp1(sfan(:,j,d),U(:,j,d),s,'linear',0);
    end
end

%=========================================================%
%      Reconstruct Image using Backprojection             %
%=========================================================%
ndim = ndet;
xb = linspace(-0.98,0.98,ndim);
yb = linspace(-0.98,0.98,ndim);
[XB,YB]=meshgrid(xb,yb);
```

```
BP = zeros(ndim,ndim);
for d=1:numdet
    BP(:,:) = BP+backproject(G(:,:,d),s,phi,xb,yb);
end

%=========================================================%
%                    Local Grids                          %
%=========================================================%
                          % Local grid size for approximating
                          % the mean and standard deviation
gridsize =11;
                          % "adgrid" - increase the local grid
                          % with 2*adgrid points to compute
                          % local std sigma. The grid for
                          % local mean stays unchanged
adgrid = 0;
k = 3;

gr = floor(gridsize/2);
Rec = zeros(size(BP));
detect =zeros(size(BP));
sigmaij = zeros(size(BP));


for i=gr+adgrid+1:ndim-gr-adgrid
  for j = gr+adgrid+1:ndim-gr-adgrid
    Reci(1:gridsize,1:gridsize) = BP(i-gr:i+gr,j-gr:j+gr);
    me = mean(Reci(:));

    Recii(1:(gridsize+2*adgrid),1:(gridsize+2*adgrid)) =...
        BP(i-gr-adgrid:i+gr+adgrid,j-gr-adgrid:j+gr+adgrid);
    sigma  = sqrt(mean(Recii(:)));
    sigmaij(i,j) = sigma;
                          % detect = number of sigmas
    detect(i,j) = (BP(i,j)-me)/sigma;
    Recold(i,j)=max([0,BP(i,j)-me-k*sigma]);
    clear me sigma_old Reci
  end
end
                           % Detect where the maximum number
                           % of sigmas occur
dx = xb(2)-xb(1); dy = yb(2)-yb(1);
[kk,ind] = max(detect(:))
```

```
[row,col] = ind2sub(size(detect),ind);
xpos = yb(1)+(col-1)*dy
ypos = xb(1)+(row-1)*dx
                              % Threshold
k_t = 4.3;

                              % Confidence based on threshold
r = 0.5*erfc(k_t/sqrt(2));
npixels  = (ndet-1)^2;
conf = (1-r)^npixels;

%=============================================================%
%              Graphs                                         %
%=============================================================%
graphpointsource
%-------------------------------------------------------------
disp('Pointsource reconstruction done !!!')
```

Below are the codes used for phantom genetarion.

Listing IV.19 'analyticphantom.m'

```
function [C U] = analyticphantom(x0,y0,r,a,alpha,psi,beta,x1
    ,x2)
% Creates analytic Cone (C) and Radon (U) data for a phantom
% consisting of disks with constant density. The vecors "x0"
% and "y0" contain the (x,y) - coordinates cenetrs of the
% disks, the vector "r" contains the radii of the discs and
% "a" gives the amplitudes. "x0", "y0", "r" and "a" must
% have the same length. The rest of the input parameters are
% as described in configure.m.
disp('Computing projections ...')
[X,Y] = meshgrid(x1(:,1),x2(:,2));
phan = zeros(length(x1(:,1)), length(x2(:,2)));
for i =1:length(x0)
     phan =  phan + a(i)*ball(x0(i),y0(i),r(i),X,Y);
end
color_plot(X,Y,phan)
view(0,90)
pause

ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);
```

```
U = zeros(ndet,nalpha,4);
C = zeros(ndet,nbeta,npsi,4);

for d=1:4
   for i = 1:length(x0)
    C(:,:,:,d) = C(:,:,:,d) + ...
    a(i)*acone(x0(i),y0(i),r(i),x1(:,d),x2(:,d),beta,psi);
    U(:,:,d) = U(:,:,d) +...
    a(i)*afanbeam(x0(i),y0(i),r(i),x1(:,d),x2(:,d),alpha);
   end
end
```

Listing IV.20 'afanbeam.m'

```
function[U] = afanbeam(x0,y0,r0,det_x,det_y,alpha)
% function[U] = afanbeam(x0,y0,r0,det_x,det_y,alpha)
% finds the exact integral (analytic projection) of the disc
% B((x0,y0),r0), centered at ('x0','y0') and with radius
% 'r0', along the lines staring at the points (det_x,det_y),
% (called detectors) and with slopes tan('alpha').
% The function returns 0 if 'alpha' is not in [0,pi].
%
%
%                 ---
%               /     \
%              |   *   |/
%               \---//
%                   /
%                  /
%                 /  ) alpha
%      ----------*-------------
%          (det_x,det_y)
%
%
% Input:
% ------------------------------------------------
% 'x0', 'y0', 'r0' are scalars defining the circular
% inclusion phantom B((x0,y0),r0).
%
% 'det_x' and 'det_y' must be vectors of same length.
% The points (det_x,det_y) must not belong to the disc
%
```

```
%'det_x','det_y' are vectors of same length conatining the
% x- and  y-coordinates of the detectors.
%
% 'alpha' is a scalar or vector of angles, measured from the
% positive x-axis and given in radians, in [0,2*pi].
%
% Output:
% ------------------------------------------
% If 'alpha' is a scalar, afanbeam returns a column
% vector U. The i-th element of U is the integral along the
% line passing through the point [det_x(i),det_y(i)] and
% with slope tan('alpha').
% If 'alpha' is a vector, the method returns an array of
% dimension length('det_x') x length('alpha'). The j-th
% column corresponds to the angle 'alpha'(j), i-th row
% corresponds to the detector ['det_x'(i),'det_y'(i)].

for j=1:length(alpha)
    for i=1:length(det_x)

        if (norm([x0-det_x(i),y0-det_y(i)])<r0 )
          disp('Detectors cannot belong to the disc B([x0,y0
             ],r0)')
          return
        end
                    % distance between the center of the
                    % disc and the detector
        c = norm([x0-det_x(i),y0-det_y(i)]);
        if det_y(i)<=y0
           a = x0-det_x(i);
        else
           a = det_x(i)-x0;
        end
        gamma1 = acos(a/c);
                     % angle b/w the line of integration and
                     % the line connecting the center of the
                     % circle to the detector
        gamma2 = abs(gamma1-alpha(j));

        if ((alpha(j) < 0)||(alpha(j) > pi))
            U(i,j)=0;
        else
         h = c*sin(gamma2);
        if (h>r0)
```

```
            U(i,j)=0;
        else
            U(i,j) = 2*sqrt(r0^2-h^2);
        end
        end
    end
end
```

Listing IV.21 'acone.m'

```
function[C] = acone(x0,y0,r0,det_x,det_y,beta,psi)
%
% function[C] = acompton2D(x0,y0,r0,det,theta,N)
% computes the analytic 2D conical transform of a
% phantom = characteristic function of the ball with center
% ('x0','y0') and radius 'r0'.
%
% Output:
% ----------------
% C is a three dimensional array with dimensions
% (length('det_x'),length('beta'),length('psi')).
% C(i,:,:) is the data for a fixed detector
% C(i,j,:) - data for fixed detector, fixed cone axis
% (and varying half-angles).
%
% Input:
% ----------------
% 'x0', 'y0','r0' describe the circular phantom and
% must be scalars.
%
% 'beta' is a vector containing the angular coordinates of
% the central axes of the cones.
%
% The apexes of the cones (i.e. detectors) have cartesian
% coordinates (x1_i,x2_i) =('det_x'(i), 'det_y'(i)), where
% 'det_x' and 'det_y' are vectors of same length.
% The detectors (det_x,det_y) must not belong to the disc
% with center ('x0','y0') and radius 'r0'.
%
% 'psi' is a vector of half-angles of cones. 'psi' must be
% uniformly distributed in [0,pi]

for i = 1:length(det_x) % loop on detectors
```

```
  for j = 1:length(beta) % loop on cone axes, i.e. on beta
    for k = 1:length(psi) % loop on half angles psi
      C(i,j,k)= ...
      afanbeam(x0,y0,r0,det_x(i),det_y(i),beta(j)-psi(k))
          +...
      afanbeam(x0,y0,r0,det_x(i),det_y(i),beta(j)+psi(k));
    end
  end
end
```

Listing IV.22 'ball.m'

```
function [U] = ball(xo,yo,r,X,Y)
% creates phantom = characteristic function of the ball B([
    xo,yo],r)
% [X,Y]-the mesh on which the function is defined, should be
    produced
% by meshgrid

%[X,Y]=meshgrid(linspace(-d, d, k));
U = double((X-xo).^2+(Y-yo).^2<=r^2);
```

Listing IV.23 'sourcephantom.m'

```
function [C, U, sourcepart, bkgdp] = sourcephantom(S,bkgd,
    ratio,alpha,psi,beta,x1,x2)
% Simulates data detected by Compton and Radon type
% emission detectors, where the phantom consists of several
% pointsources and background radiation.
%
% Output:
% ---------
% C - Compton data, array of dimension
% length(x1)xlength(beta)xlength(psi)x4,
% U - Radon data, array of dimension
% length(x1)xlength(alpha)x4.
% "sourcep" and "bkgdp" are 1x4 vector, such that
% sourcep(i) and bkgdp(i) are the number of source
% and background particles, correspondingly, detected by
% detector i.
% Input:
```

```
% ----------
% "S" is a 2xnsource array, where nsource is the number of
% point sources. S(:,i) are the (x,y)-coordinates of the
% i-th source. "bkgd" is the total number of background
% particles, "ratio" is the ratio of the source particles
% coming from ONE of the sources to the background. The rest
% of the input parameters are as described in configure.m.

disp('Computing source phantom...')
npart = ceil(bkgd*ratio);
ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);
U = zeros(ndet,nalpha,4);
C = zeros(ndet,nbeta,npsi,4);

sourcepart=zeros(1,4);
for i=1:length(S(1,:))
  fprintf('Source particles from source %1.0f \n', i)
  [Ci,Ui,sourcepi]=...
      pointsource(S(:,i)',npart,alpha,psi,beta,x1,x2);
  C = C+Ci; U=U+Ui; sourcepart = sourcepart+sourcepi;
end
[Cbkgd,Ubkgd,bkgdp] =...
    background(bkgd,alpha,psi,beta,x1,x2);

C = C+Cbkgd; U = U+Ubkgd;
```

Listing IV.24 'pointsource.m'

```
function [C, U, sourcepart] = pointsource(S,npart,alpha,psi,
   beta,x1,x2)
% Simulates data detected by Compton and Radon type
% emission detectors, placed on the sides of the unit
% square. The particles are emitted by a point source with
% coordinates (S(1),S(2)).
%
% Input:
% ----------
% S is a vector containg the two coordinates of a
% point-source.
% "npart" = number of particles coming from the source
% The rest of the input parameters are as described in
% configure.m
```

```
%
% Output:
% ---------
% "C" contains detecttion data from Compton type detectors,
% "U" contains data from Radon type detectors.
% "C" has dimensions length(x1)xlength(beta)xlength(psi)x4,
% "U" has dimensions length(x1)xlength(alpha)x4.
%
% ------------------------------------------------------------
%
% 1. A photon is emitted from the source S into a random
% direction in [0,2*pi) and hits a detector. The incoming
% direction alpha in [0,pi] is determined.
% 2. The photon then scatters, with a scattering angle "psi"
% "psi" is (uniformly) randomly chosen in (-alpha,pi-alpha).
% In other words, "psi" is the angle the new direction
% (after the scattering) forms with the original direction,
% counted from the original direction.
%
%  scat.   orig        orig    scat
%      \ang|           |-ang/
%        \ |           |   /          here ang is in (0,pi)
%         \|           |/
%          o           o
%      detector     detector
%
% 3. The central axis beta is determined by beta=alpha+psi.
% The Compton data C(i,j,k,d) contains the number of
% particles which hit detector array "d" at detector bin
% "i", scattered with an angle "psi(k)" or -"psi(k)" into
% direction "beta(j)".
% The Radon data U(i,j,d) contains the number of particles
% that hit detector array "d" at detector bin "i", from
% incoming direction "alpha(j)".


%=============================================================%
%                          Setup                              %
%=============================================================%
ndet = length(x1); nalpha = length(alpha);
npsi = length(psi); nbeta = length(beta);
U = zeros(ndet,nalpha,4);
C = zeros(ndet,nbeta,npsi,4);
                          % detc will hold the locations of
```

```
                                % hits for the 4 detecrors
detc = cell(1,4);
                                % alphac will hold the directions
                                % of incoming particles for the 4
                                % detectors
alphac = cell(1,4);
                                % Scattering angles
psic = cell(1,4);
                                % Central axes
betac = cell(1,4);
                                % Initialize rand to a different
                                % state each time
rand('state', sum(100*clock));

%=========================================================%
%              Photons Emitted and Detected               %
%=========================================================%
                              % A photon is emitted into
                              % a random direction with polar
                              % coordinate A in [0,2*pi)
A = (2*pi-0.001)*rand(1,npart);

%---------------------------------------------------------%
%                    Hit a detector                       %
%---------------------------------------------------------%
                             % Determine which detector array
                             % is hit and record the location
                             % and incoming direction of the
                             % hit.
for i = 1:npart

    if 0<A(i) & A(i)<pi                  % Go UP
%-------------------------------------------------------------
        x20 = x2(1,3);
        x10 = S(1) + (x20-S(2))*cos(A(i))/sin(A(i));

        if x1(1,3)<= x10 && x10<=x1(end,3)
                             % Detector III is hit...
                             % x10 is the exact location of the
                             % hit on this detector array
            detc{3} =[detc{3}; x10];
                             % incoming direction of the particle
            alphac{3} = [alphac{3}; A(i)];
        else
```

```
            if x10 < x1(1,3)
                        % Detector  IV is hit
                x10 = x1(1,4);
                x20 = S(2)+(x10-S(1))*sin(A(i))/cos(A(i));
                detc{4} = [detc{4}; x20];
                alphac{4} = [alphac{4}; A(i)];
            else
                        % Detector II is hit
                x10 = x1(1,2);
                x20 = S(2)+(x10-S(1))*sin(A(i))/cos(A(i));
                detc{2} = [detc{2}; x20];
                alphac{2} = [alphac{2}; A(i)];
            end
        end


    else
        if pi<A(i) && A(i)<2*pi        % Go DOWN
%-------------------------------------------------------------------------
        x20 = x2(1,1);
        x10 = S(1) + (x20-S(2))*cos(A(i))/sin(A(i));

            if x1(1,1)<= x10 && x10<=x1(end,1)
                        % Detector I is hit
               detc{1} = [detc{1}; x10];
               alphac{1} = [alphac{1}; A(i)-pi];
            else
                if x10 < x1(1,1)
                        % Detector  IV (l) is hit
                   x10 = x1(1,4);
                   x20 = S(2)+(x10-S(1))*sin(A(i))/cos(A(i))
                       ;
                   detc{4} = [detc{4}; x20];
                   alphac{4} = [alphac{4}; A(i)-pi];
                else
                        % Detector II (r) is hit
                   x10 = x1(1,2);
                   x20 = S(2)+(x10-S(1))*sin(A(i))/cos(A(i))
                       ;
                   detc{2} = [detc{2}; x20];
                   alphac{2} = [alphac{2}; A(i)-pi];
                end
            end
```

```
        else % A=0 or A=pi, i.e. go straight RIGHT or LEFT
%-------------------------------------------------------------------------
            if A(i)==0
                        % Go to detector II
                detc{2} = [detc{2}; S(2)];
                alphac{2} = [alphac{2}; A(i)];
            else
                        % A=pi, go to detector IV
                detc{4} = [detc{4}; S(2)];
                alphac{4} = [alphac{4}; A(i)];
            end
    end
 end

 end % for cycle on npart
%keyboard
%-------------------------------------------------------------------------%
%         Scattering angle psi and axis beta                              %
%-------------------------------------------------------------------------%
                    % For each alpha, the particles scatter
                    % randomly with a (signed) scattering angle
                    % in (-alpha,pi-alpha)\{0}, more precisely,
                    % in [-alpha+dpsi/2,-dpsi/2-0.0001] U
                    % [dpsi/2,pi-alpha-dpsi/2-0.0001]. Thus,
                    % particles that scatter very little are not
                    % detected. The chosen intervals guarantee
                    % that detection occurs. Note that, as we
                    % took care dpsi = dbeta, the intervals we
                    % chose for psi guarantee that
                    % beta \in [dbeta/2,pi-dbeta/2].
dpsi = psi(2)-psi(1); eps = 0.0001;
for d=1:4
    psic{d} = (pi-2*dpsi-2*eps)*rand(length(alphac{d}),1);
end
for d=1:4
    psic{d}(psic{d}<=alphac{d}-eps-dpsi)=...
        psic{d}(psic{d}<=alphac{d}-eps-dpsi)-...
           alphac{d}(psic{d}<=alphac{d}-eps-dpsi)+dpsi/2;
    psic{d}(psic{d}>alphac{d}-eps-dpsi)=...
        psic{d}(psic{d}>alphac{d}-eps-dpsi)-...
         alphac{d}(psic{d}>alphac{d}-eps-dpsi)+eps+3*dpsi/2;
end
                    % Determine the cenrtal axes of cones 'beta'
```

```
                % beta = alpha + psi
for d=1:4
    betac{d} = alphac{d}+psic{d};
end
                % Take the absolute value of 'psi', becuase
                % compton data only 'knows' cos(psi),
                % i.e. |psi|.
for d=1:4
psic{d} = abs(psic{d});
end

%=========================================================%
%                    Binning                              %
%=========================================================%
deti = cell(1,4); alphai = cell(1,4);
psii = cell(1,4); betai = cell(1,4);
                % Sizes of bins of each detector array
dx(1) = x1(2,1)-x1(1,1); dx(2) = x2(2,2)-x2(1,2);
dx(3) = x1(2,3)-x1(1,3); dx(4) = x2(2,4)-x2(1,4);
                % Binning of detectors
for d=[1 3]
    deti{d} = round( (detc{d}-x1(1,d))/dx(d) )+1;
end
for d=[2 4]
    deti{d} = round( (detc{d}-x2(1,d))/dx(d) )+1;
end
                % Binning of incoming directions alpha
                % This is for Radon data only
dalpha = alpha(2)-alpha(1);
for d=1:4
    alphai{d} = round(alphac{d}/dalpha)+1;
end
                % Binning of scattering angles psi
for d=1:4
    psii{d} = round(psic{d}/dpsi);
end
                % Binning of central axes beta
dbeta =  beta(2)-beta(1);
for d=1:4
    betai{d} = round(betac{d}/dbeta);
end

%=========================================================%
%            Fill Compton and Radon arrays                %
```

```
%=========================================================================%
C = zeros(ndet,nbeta,npsi,4);
U = zeros(ndet,nalpha,4);
%keyboard
for d=1:4
    for i=1:length(deti{d})
        C(deti{d}(i),betai{d}(i),psii{d}(i),d) = ...
            C(deti{d}(i),betai{d}(i),psii{d}(i),d)+1;
        U(deti{d}(i),alphai{d}(i),d) = ...
            U(deti{d}(i),alphai{d}(i),d) +1;
    end
end



for d=1:4
    sourcepart(d) = sum(sum(sum(C(:,:,:,d))));
end

fprintf('%1.0f source particles hit detector array I \n',...
    sourcepart(1) )
fprintf('%1.0f source particles hit detector array II \n'
    ,...
    sourcepart(2) )
fprintf('%1.0f source particles hit detector array III \n'
    ,...
    sourcepart(3) )
fprintf('%1.0f source particles hit detector array IV \n'
    ,...
    sourcepart(4) )
fprintf('%1.0f TOTAL SOURCE particles \n\n',sum(sourcepart))
```

Listing IV.25 'background.m'

```
function [Cbkgd,Ubkgd,bkgdpart] = background(bkgd,alpha,psi,
   beta,x1,x2)
% Uniform background for Compton and Radon data,
% collected by four detector arrays lying on the sides of
% the unit square

bkgd4 = round(bkgd/4);
                % Initialize rand to a different state
                % each time
```

```
rand('state', sum(100*clock));

%=========================================================================%
%                    Data collection                                      %
%=========================================================================%

%-------------------------------------------------------------------------%
%         Hit a detector                                                  %
%-------------------------------------------------------------------------%
                    % Particles hit the detectors.
det(:,1) = x1(1,1)+(x1(end,1)-x1(1,1))*rand(bkgd4,1);
det(:,2) = x2(1,2)+(x2(end,2)-x2(1,2))*rand(bkgd4,1);
det(:,3) = x1(1,3)+(x1(end,3)-x1(1,3))*rand(bkgd4,1);
det(:,4) = x2(1,4)+(x2(end,4)-x2(1,4))*rand(bkgd4,1);

%-------------------------------------------------------------------------%
%         Directions of the incoming partilces                            %
%-------------------------------------------------------------------------%
                    % Choose directions of incoming particles
                    % 'alpha' uniformly distributed in the
                    % interval [0,pi].
dalpha = alpha(2)-alpha(1);
alphac = pi*rand(bkgd4,4);

                    % For each alpha, the particles scatter
                    % randomly with a (signed) scattering angle
                    % in (-alpha,pi-alpha)\{0}, more precisely,
                    % in [-alpha+dpsi/2,-dpsi/2-0.0001] U
                    % [dpsi/2,pi-alpha-dpsi/2-0.0001]. Thus,
                    % particles that scatter very little are not
                    % detected. The chosen intervals guarantee
                    % that detection occurs. Note that, as we
                    % took care dpsi = dbeta, the intervals we
                    % chose for psi guarantee that
                    % beta \in [dbeta/2,pi-dbeta/2].
dpsi = psi(2)-psi(1); eps = 0.0001;
psic = (pi-2*dpsi-2*eps)*rand(bkgd4,4);
for d=1:4
psic(psic(:,d)<=alphac(:,d)-eps-dpsi,d)=...
     psic(psic(:,d)<=alphac(:,d)-eps-dpsi,d)-...
          alphac(psic(:,d)<=alphac(:,d)-eps-dpsi,d)+dpsi/2;
psic(psic(:,d)>alphac(:,d)-eps-dpsi,d)=...
     psic(psic(:,d)>alphac(:,d)-eps-dpsi,d)-...
       alphac(psic(:,d)>alphac(:,d)-eps-dpsi,d)+eps+3*dpsi/2;
```

```
end
                % Determine the cenrtal axes of cones 'beta'
                % beta = alpha + psi
dbeta =  beta(2)-beta(1);
betac = alphac+psic;
                % Take the absolute value of 'psi', becuase
                % compton data only 'knows' cos(psi),
                % i.e. |psi|.
psica = abs(psic);

%=========================================================%
%                    Binning                              %
%=========================================================%
                % Sizes of bins of each detector array
dx(1) = x1(2,1)-x1(1,1); dx(2) = x2(2,2)-x2(1,2);
dx(3) = x1(2,3)-x1(1,3); dx(4) = x2(2,4)-x2(1,4);
                % Binning of detectors
det_ind(:,1) = round( (det(:,1)-x1(1,1))/dx(1) )+1;
det_ind(:,2) = round( (det(:,2)-x2(1,2))/dx(2) )+1;
det_ind(:,3) = round( (det(:,3)-x1(1,3))/dx(3) )+1;
det_ind(:,4) = round( (det(:,4)-x2(1,4))/dx(4) )+1;
                % Binning of incoming directions alpha
                % This is for Radon data only
alpha_ind = round(alphac/dalpha)+1;
                % Binning of central axes beta
beta_ind = round(betac/dbeta);
                % Binning of opening angles psi
psi_ind = round(psica/dpsi);

%=========================================================%
%             Fill Compton and Radon arrays               %
%=========================================================%
ndet = length(x1(:,1)); nalpha = length(alpha);
nbeta = length(beta); npsi = length(psi);

Ubkgd = zeros(ndet,nalpha,4);
Cbkgd = zeros(ndet,nbeta,npsi,4);
for d = 1:4
    for i=1:round(bkgd/4)
        Cbkgd(det_ind(i,d),beta_ind(i,d),psi_ind(i,d),d) = ...
            Cbkgd(det_ind(i,d),beta_ind(i,d),psi_ind(i,d),d)+1;
        Ubkgd(det_ind(i,d),alpha_ind(i,d),d) = ...
            Ubkgd(det_ind(i,d),alpha_ind(i,d),d) +1;
    end
```

```
end


for d=1:4
    bkgdpart(d) = sum(sum(sum(Cbkgd(:,:,:,d))));
end

fprintf('%1.0f bkgd particles hit detector array I\n',...
    bkgdpart(1))
fprintf('%1.0f bkgd particles hit detector array II\n',...
    bkgdpart(2))
fprintf('%1.0f bkgd particles hit detector array III \n',...
    bkgdpart(3))
fprintf('%1.0f bkgd particles hit detector array IV \n',...
    bkgdpart(4))
fprintf('%1.0f TOTAL BACKGROUND particles \n',sum(bkgdpart))
fprintf('%1.0f background particles detected by detector
   arrays I, II and III \n\n',bkgdpart(1)+bkgdpart(2)+
   bkgdpart(3))
```

The next two listings are of the implementations of methods B and C.

Listing IV.26 'cone2fanbeamB.m'

```
function [U] = cone2fanbeamB(C)
% Given compton data C find the Radon data in fanbeam
% coordinates, U = Rf((x1,x2),alpha). For a fixed direction
% alpha0, Rf((x1,x2),alpha0) is found by manipulating data
% from three Compton cones. Averaging on such cones is used,
% so which results in theuse of all available data and
% decreases noise.
%
% Output:
% ----------
% U is an array of dimension ndet x nalpha, where
% ndet = length(C(:,1,1)), nalpha = (lengthC(1,:,1)+2).
% This is consistent with the setup defined in configure.m,
% namely, the angles alpha change in [0,pi], while
% beta and psi are in (0,pi). alpha, beta, psi are
% partitioned with the same step size, but alpha has 2 more
% elements, corresponding to 0 and pi.
%
% Input:
```

```
% ------------
% "C" is a Compton data array of dimension
% ndet x nbeta x npsi.
fprintf('starting cone2fanbeam_v2...')
tic

ndet = length(C(:,1,1));
nbeta = length(C(1,:,1));
npsi = length(C(1,1,:));
nalpha = npsi+2;

psi_index = [-npsi:-1,1:npsi];

U = zeros(ndet,nalpha);

for i = 1:ndet % loop on detectors
    i
  for j = 1:nalpha % loop on alpha
    c=1;
    for k = psi_index % loop on psi_k
      for l= psi_index % loop on psi_l
        condition = ...
        (1<=j+k-1)&(j+k-1<=nbeta)&...
        (1<=j+l-1)&(j+l-1<=nbeta)&...
        (1<=j+k+l-1)&(j+k+l-1<=nbeta)&...
        (1<=abs(k-l))&(abs(k-l)<=nbeta);
        if condition
        U(i,j) = ( C(i,j+k-1,abs(k))+C(i,j+l-1,abs(l)) -...
                   C(i,j+k+l-1,abs(k-l)) )/2+U(i,j);
            c=c+1;
        end % end if condtion
        end % psi_l
    end %psi_k
  U(i,j)= U(i,j)/(c-1);%nalpha*U(i,j)/c; %c+1?
  end %alpha
end %detectors
fprintf('time = %4.2f \n',toc)
```

Listing IV.27 'cone2fanbeamC.m'

```
function[U] = cone2fanbeamC(C,phan)
% Given compton data C find the Radon data in fanbeam
% coordinates, U = Rf((x1,x2),alpha). For a fixed direction
```

```
% alpha0, Rf((x1,x2),alpha0) is approximated by the average
% of all cones that have one side parallel to alpha0.
%
% Output:
% ----------
% U is an array of dimension ndet x nalpha, where
% ndet = length(C(:,1,1)), nalpha = (lengthC(1,:,1)+2).
% This is consistent with the setup defined in configure.m,
% namely, the angles alpha change in [0,pi], while
% beta and psi are in (0,pi). alpha, beta, psi are
% partitioned with the same step size, but alpha has 2 more
% elements, corresponding to 0 and pi.
%
% Input:
% ----------
% "C" is a Compton data array of dimension
% ndet x nbeta x npsi.
% If "phan" is 's', then U((x1,x2),alpha0) is not the
% average of cones with side || alpha0, but the sum of these
% cones. When "pointsource" is not 1, an average of this sum
% is taken.
fprintf('starting cone2fanbeamC...  ')
tic
Ndet = length(C(:,1,1));
Nbeta = length(C(1,:,1));
Npsi = length(C(1,1,:));
Nalpha = Nbeta+2;

psi_index = [-Npsi:-1,1:Npsi];

U = zeros(Ndet,Nalpha);

for i = 1:Ndet % loop on detectors
    for j = 1:Nalpha % loop on alpha
        c=0;
        for k = psi_index % loop on psi_k
            if (1<=j+k-1)&(j+k-1<=Nbeta)
             U(i,j) = U(i,j) + C(i,j+k-1,abs(k));
              c=c+1;
            end
        end %psi_k
        if phan ~='s'
            U(i,j) = U(i,j)/c;
        end
```

```
    end %alpha
end %detectors
fprintf('time = %4.2f \n',toc)
```

The auxiliary codes follow.

Listing IV.28 'radon_inversion.m'

```
function [FBP] = radon_inversion(G,s,theta,xb,yb,filter)
% Inverts the 2D Radon (X-ray) transform using filtered
% backprojection.
%
% Input:
%-----------
% 'G' - Radon data; a matrix in which every column
% corresponds to one projection ('theta'(j)) and conatins
% the values of the line integrals of F, parametrized by S.
% 'theta' - a vector of projection angles.
% 's' - lines in each projection ('s' - distances of the
% lines form the origin).
% 'xb','yb' are vectors containing the coordinates of the
% points where the reconstructed image is to be found.
% 'filter' - an integer which determines the type of filter
% to be used: 1 -> FBP filter H*d/ds, 2 -> modified FBP
% filters with zero frequency cut, 3 -> modified FBP with
% high frequencies killed, 4-> local tomography d^2/ds^2,
% 5->local tomo filter with high frequencies killed,
% 7-> local tomo filter implemented using finite differences
% 6-> backprojection, 8-> backprojection with high
% frequencies cut.
%
% Output:
%-----------
% Returns the matrix FBP of the filtered backprojection
% reconstruction. FBP has dimensions
% [length(yb),length(xb)].
fprintf('Starting radon inversion... ')
tic
nlines = length(s); nproj = length(theta);
a = max(s)-min(s);

%=========================================================%
```

```
%                        Filtration                                    %
%======================================================================%

%----------------------------------------------------------------------%
%     Regular filter: H*d/ds          filters 1,2,3                    %
%----------------------------------------------------------------------%
switch filter
  case  {1,2,3} %'fbp' filter /\
   fprintf('filter %d : FBP filter |sigma| ', filter)
   pad = 2*nlines;
   mid = round((nlines+pad)/2);
                          % Smoothly pad every column of G
                          % with "pad" zeros at the end and
                          % then take FFT of each column
                          % (i.e. row-wise), on "s" variable
    k0 = 14;
    smooth=[0.5+0.5*cos((0:k0-1)*pi/k0),zeros(1, pad/2-k0)];
    sm_r = smooth'*G(nlines,:);
    sm_l = flipud(smooth')*G(1,:);
    FG = [G;sm_r;sm_l];
    FG = fft(FG);
                          % Flt is a /\ filter in frequency
                          % domain.
    Flt = (2*pi/a)*nlines*...
                      linspace(0,1,ceil((nlines+pad)/2)+1);
    Flt = [Flt Flt(end-1:-1:2)];

    if (filter == 2)
     % fbp filter with smoothly cut 0 frequency
     disp('with 0 frequency killed')
                          % cut 0 frequency smoothly
                          % filter = 0 when |sigma|<=k0,
                          % filter is smooth when
                          % k0<|sigma| <=k1
                          % otherwise filter = |sigma|
     k1 = round(length(Flt)*1/100);
     k2=round(length(Flt)*5/100);
     k = (k1+1):(k1+k2);
     sm0 = [zeros(1,k1),0.5-cos((k-k1)*pi/k2)*0.5,...
                        ones(size(k1+k2+1:(nlines+pad)/2))];
     sm0 = [sm0,fliplr(sm0)];
     Flt = Flt.*sm0;
    end
```

```
    if (filter ==3)
      % fbp filter with smoothly cut high frequences
      disp(' with high freqiencies cut')
      k3 = round(mid*5/100);
      k4 = round(mid*10/100);
      kk = mid-k4:mid-k3;
      c = 0;
      Flt(mid-k3:mid+k3) = c;
      smh1 = (Flt(mid-k4)-c)*...
                    (1+cos((kk-mid+k4)*pi/(k4-k3)))/2+c;
      smh2 = fliplr((Flt(mid+k4)-c)*...
                    (1+cos((kk-mid+k4)*pi/(k4-k3)))/2+c);
      Flt(kk) = smh1; Flt(kk+k3+k4) = smh2;
    end

%-------------------------------------------------------------------%
%  Local Tomography filters d^2/ds^2    filters 4,5 and 7  %
%-------------------------------------------------------------------%
  case {4,5} % Filter implemented using FFT
    fprintf('filter %d : local filter using FFT',filter)
    FG = fft(G);
    % Flt = (2*pi/a)*nlines*linspace(0, 1, nlines/2+1);
    % Flt = [Flt Flt(end-1:-1:2)].^2;
    abs_sigma = abs((1:nlines)-nlines/2);
    Flt = fftshift(abs_sigma.^2);

    if filter == 4 % high frequences cut-off
      fprintf(' and high frequences cut-off')
      mid=round(nlines/2);
      k3 = round(mid*70/100); k4 = round(mid*75/100);
      kk = mid-k4:mid-k3; c = 0;
      Flt(mid-k3:mid+k3) = c;
      smh1 = (Flt(mid-k4)-c)*...
                  ( 1+cos((kk-mid+k4)*pi/(k4-k3)) )/2+c;
      smh2 = fliplr((Flt(mid+k4)-c)*...
                    (1+cos((kk-mid+k4)*pi/(k4-k3)))/2+c);
      Flt(kk) = smh1; Flt(kk+k3+k4) = smh2;
    end

  case 7
  % Filter implemented using finite differences -> smoothing
    fprintf('filter %d : Local filter using finite
       differences', filter)
    FltG = zeros(size(G));
```

```
    ds = a/(nlines-1);
    for i=1:length(theta)
      for j = 2:length(s)-1
       FltG(j,i) = -(G(j+1,i) - 2.0*G(j,i)...
                                        + G(j-1,i) )/(ds*ds);
      end
    end
%-------------------------------------------------------------%
%           Backprojection  filters 6,8                       %
%-------------------------------------------------------------%
  case 6  % No filter, only backprojection
    fprintf('filter %d : Only backprojection', filter)
    FltG = G;

  case 8 %backprojection with high frequences cut
    fprintf('filter %d : backprojection with high
       frequencies cut',filter)
    FG = fft(G);
    mid=round(nlines/2);
    k0 = round(mid*70/100);
    k1 = round(mid*75/100);
    Flt = ones(1,nlines);
    kk = mid-k1:mid-k0;
    c = 0;%Flt(mid-k3);
    Flt(mid-k0:mid+k0) = c;
    smh1 = (Flt(mid-k1)-c)*...
                        (1+cos((kk-mid+k1)*pi/(k1-k0)))/2+c;
    smh2 = fliplr((Flt(mid+k1)-c)*...
                       (1+cos((kk-mid+k1)*pi/(k1-k0)))/2+c);
    Flt(kk) = smh1; Flt(kk+k0+k1) = smh2;
end % end switch

%-------------------------------------------------------------%
%                      Apply Filter                           %
%-------------------------------------------------------------%
if (filter <=5| filter == 8)

                    % convert filter to a matrix in order to
                    % avoid a loop on columns when applying
                    % filter
  Flt = Flt'*ones(1,nproj);
                    % Apply filter
  FltG = FG.*Flt;
                    % Take inverse FFT
```

```
  FltG = ifft(FltG);
                    % Cut irrelevant values (if any)
  FltG = real(FltG(1:nlines,:));

end

%=========================================================%
%                    Backprojection                       %
%=========================================================%
ndimx = length(xb);ndimy = length(yb);
stepx = xb(2)-xb(1); stepy = yb(2)-yb(1);
anglestep = theta(2)-theta(1);
sstep = s(2)-s(1);
FBP = zeros(ndimy,ndimx);

for ix = 1:ndimx
  x = xb(ix);
  for iy = 1:ndimy
    y = yb(iy);
    if( x*x + y*y < (a/2)^2 - 0.001)

    for npr = 1:nproj
      angle = theta(1) + (npr-1)*anglestep;
      co = cos(angle); si = sin(angle);
      s = x*co + y*si;

      position = (nlines+1)/2 + s/sstep ;
      nline0 = floor(position);
      nline1 = nline0 + 1;
      alpha1 = position - nline0;
      alpha0 = 1 - alpha1;

      if(alpha1>1 | alpha1<0 | nline1>nlines | nline0<1)
         display('linear interpolation is not working');
         return
      end

      value=alpha0*FltG(nline0,npr)+alpha1*FltG(nline1,npr);

      trapweight = anglestep;
      if(npr == 1 | npr == nproj)
        trapweight = 0.5*anglestep;
      end

      FBP(iy,ix) = FBP(iy,ix) + value*trapweight /(4.0*pi);
```

```
    end % for npr
    end %if
  end % for iy
end % for ix
fprintf('\n time = %4.2f \n', toc)
```

Listing IV.29 'backproject.m'

```
function [BP] = backproject(G,s,theta,xb,yb)

fprintf('Starting backprojection... ')
tic
nlines = length(s); nproj = length(theta);
a = max(s)-min(s);

ndimx = length(xb);ndimy = length(yb);
stepx      =  xb(2)-xb(1);
stepy      =  yb(2)-yb(1);
anglestep  = theta(2)-theta(1);
sstep      = a/(nlines-1);

BP = zeros(ndimy,ndimx);

for ix = 1:ndimx
  x = xb(ix);
  for iy = 1:ndimy
      y = yb(iy);
      if( x*x + y*y < (a/2)^2 - 0.001)
        for npr = 1:nproj
         angle = theta(1) + (npr-1)*anglestep;
         co = cos(angle);
         si = sin(angle);
         s = x*co + y*si;
         position = (nlines+1)/2 + s/sstep;
         nline0 = floor(position);
         nline1 = nline0 + 1;
         alpha1 = position - nline0;
         alpha0 = 1 - alpha1;

         if(alpha1>1 | alpha1<0 | nline1>nlines | nline0<1)
           display('linear interpolation is not working');
           return
         end % if
```

```
            value = alpha0*G(nline0,npr)+alpha1*G(nline1,npr);

            trapweight = anglestep;
            if(npr == 1 | npr == nproj)
              trapweight = 0.5*anglestep;
            end % if

            BP(iy,ix) = BP(iy,ix) + value;

          end % for
        end %if
      end % for
end % for

fprintf(' time = %4.2f \n', toc)
```

Listing IV.30 'graphrec.m'

```
if phan == 'a'  % reconstruction of analytic phantom
 for i=filter
  Rec = squeeze(FBP(:,:,i));
  figure(2*i-1)
  subplot(1,2,1);
  gray_plot(XB,YB,exact)
  view(0,90)
  title('Phantom')
  subplot(1,2,2)
  gray_plot(XB,YB,Rec); view(0,90)
  title(['Reconstruction using filter ', num2str(i)])
  hold on
  for d=1:numdet
  plot3(x1(:,d),x2(:,d),0*ones(1,length(x1(:,d))),'*g',...
                                                  'LineWidth',2)
  end
  hold off

  %---------------------- Profile plots at y = yi and x = xi
    yi = c(2,1);
    xi = c(1,1);
    hr = xb(2)-xb(1);
                          % 'at' determines the entries of
                          % the arrays corresponding to 'y0'
```

```
    aty =floor((1.1+yi)/hr);
    atx = floor((1.1+xi)/hr);
    figure(2*i)
    subplot(2,1,1)
    plot(xb,Rec(aty,:),'-k','LineWidth',2)
    axis([-1 1 -0.1, 1.2])
    hold on
    plot(xb,exact(aty,:),':k','LineWidth',2)
    xlabel('x')
    legend('Reconstruction','Phantom')
    title(['y =',num2str(yi),' profile' ])
    hold off
    subplot(2,1,2)
    plot(yb,Rec(:,atx),'-k','LineWidth',2)
    axis([-1 1 -0.1, 1.2])
    hold on
    plot(yb,exact(:,atx),':k','LineWidth',2)
    xlabel('y')
    legend('Reconstruction','Phantom')
    title(['x =',num2str(xi),' profile'])
    hold off
 end
else        % reconstruction of source phantom
 for i=filter
  Rec = squeeze(FBP(:,:,i));
  me = mean(Rec(:));
  sigma = std(Rec(:));
  Rec_cut = (Rec-me)/sigma;
  Rec_cut4 = Rec_cut.*(Rec_cut>4);
  ang = 0:0.01:2*pi;
  sourcep = 0; bkgdp=0;
  for d=1:numdet
    sourcep = sourcep + sourcepart(d);
    bkgdp = bkgdp + bkgdpart(d);
  end

  datastr =' ';
  if data == 'c'
     datastr = 'Compton'
  elseif data == 'r'
     datastr = 'Radon'
  end

  figure(i)
  color_plot(XB,YB,Rec);
```

```
    title({[datastr,', ',num2str(numdet),' detectors, '];...
    [num2str(sourcepart), ' source particles / ',...
    num2str(bkgdpart),' background particles, Source at (',...
    num2str(S(1)),',',num2str(S(2)),')'];...
    ['Reconstruction by FBP,  filter ', num2str(i)]},...
    'FontSize',14)
    hold on
    for d=1:numdet
     plot3(x1(:,d),x2(:,d),0*ones(1,length(x1(:,d))),'*k',...
                                                  'LineWidth',2)
    end
    hold off

    figure(i+1)
    color_plot(XB,YB,Rec_cut4);
    view(0,90)
    title({[datastr,', ',num2str(numdet),' detectors, '];...
    [num2str(sourcepart), ' source particles / ',...
    num2str(bkgdpart),' background particles, Source at (',...
    num2str(S(1)),',',num2str(S(2)),')'];...
    ['Reconstruction by FBP, above mean+4\sigma, filter ',...
    num2str(i)]},'FontSize',14)
    zlabel('\sigma')
    hold on
    for d=1:numdet
     plot3(x1(:,d),x2(:,d),0*ones(1,length(x1(:,d))),'*k',...
                                                  'LineWidth',2)
    end
    plot3(0.05*cos(ang)+S(1),0.05*sin(ang)+S(2),...
                  0*ones(1,length(ang)),'b','LineWidth',2)
    hold off

      end % for
end % if pointsource
```

Listing IV.31 'graphpointsource.m'

```
% Graphs for pointsource reconstructions via backprojection

sourcep = 0; bkgdp=0;
for d=1:numdet
     sourcep = sourcep + sourcepart(d);
     bkgdp = bkgdp + bkgdpart(d);
```

```
end

datastr ='';
if data == 'c'
    datastr = 'Compton';
elseif data == 'r'
    datastr = 'Radon';
end

detect_cut = detect;
detect_cut(detect_cut<=k_t)=0;

figure(1)
color_plot(XB,YB,detect_cut);
title({[datastr,', ', num2str(numdet),' detectors'];...
 ['Loc mean on ',num2str(gridsize),'x',num2str(gridsize),...
 'grid, Loc \sigma on ', num2str(gridsize+2*adgrid),'x',...
 num2str(gridsize+2*adgrid),' grid, max num loc \sigma ',...
 num2str(kk)]; ['Source detected at (',num2str(xpos),',',...
 num2str(ypos),'), confidence ',num2str(100*conf),'%'];
 ['Plot of (Reconstruction - mean-',num2str(k_t),...
 '\sigma)/ \sigma']},'FontSize',14)
hold on
for d=1:numdet
 plot3(x1(:,d),x2(:,d),0*ones(1,length(x1(:,d))),'*k',...
                                                 'LineWidth',2)
end
hold off

figure(2)
meandetect = mean(detect(:));
color_plot(XB,YB,detect);
title({[datastr,', ', num2str(numdet),' detectors'];...
 ['Loc mean on ',num2str(gridsize),'x',num2str(gridsize),...
 'grid, Loc \sigma on ', num2str(gridsize+2*adgrid),'x',...
 num2str(gridsize+2*adgrid),' grid, max num loc \sigma ',...
 num2str(kk)];['Source detected at (',num2str(xpos),',',...
 num2str(ypos),'), confidence ',num2str(100*conf),'%'];
 ['Plot of (Reconstruction - mean)/ \sigma']},'FontSize',14)
hold on
for d=1:numdet
plot3(x1(:,d),x2(:,d),meandetect*ones(1,length(x1(:,d))),...
                                          '*k','LineWidth',2)
end
hold off
```

```
figure(3)
color_plot(XB,YB,BP);
title({[datastr,', ', num2str(numdet),...
   ' detectors, Source at (',num2str(S(1)),',',...
  num2str(S(2)),')' ]; [num2str(sourcep),...
   ' source particles / ', num2str(bkgdp),...
   ' background particles'];...
   ['Reconstruction by Backpojection']},'FontSize',14)
```

Listing IV.32 'color_plot.m'

```
function [] = color_plot(X,Y,U)
% creates a "color plot" of U
Uplot = U;
surf(X,Y,Uplot)
colorbar
colormap(jet)
view(10,90)
axis([-1.1 1.1 -1.1 1.1 min(min(U)) max(max(U))+0.01])
shading interp
```

Listing IV.33 'gray_plot.m'

```
function [] = gray_plot(X,Y,U)
Uplot = U;
surf(X,Y,Uplot)
colorbar
colormap(gray)
view(0,90)
axis([X(1,1) X(end,end) Y(1,1) Y(end,end) min(min(U)) max(
   max(U))+0.01])
xlabel('x')
ylabel('y')
shading interp
```

VITA

Yulia Nekova Georgieva-Hristova was born in Sofia, Bulgaria. She studied at Sofia University "St. Kliment Ohridski" from September 1998 and graduated in June 2002 with a Bachelor of Science degree in Mathematics. During 2003 Yulia studied in the graduate program at the Department of Mathematics at Politecnico di Torino, Italy. She enrolled in the doctoral program in the Department of Mathematics, Texas A&M University in the fall of 2004. The current dissertation on mathematical problems of thermoacoustic and Compton camera imaging was defended in June 2010. Yulia can be contacted at:

IMA, University of Minnesota

114 Lind Hall,

Minneapolis, MN 55455

Publication List

- M. Allmaras, D. Darrow, Y. Hristova and P. Kuchment, *Detecting small low emission radiating sources* (in preparation)
- Y. Hristova, *Time reversal in thermoacoustic tomography - an error estimate*, Inverse Problems 25 (2009) 055008
- Y. Hristova, P. Kuchment, L. Nguyen *On reconstruction and time reversal in thermoacoustic tomography in homogeneous and non-homogeneous acoustic media*, Inverse Problems 24 (2008)

The typist for this dissertation was Yulia Nekova Georgieva-Hristova.