Hybrid Methods for Large-Scale Tomographic Reconstructions

Ditte Iben Marcussen

Kongens Lyngby 2012
IMM-M.Sc.-2012-16

Technical University of Denmark
DTU Informatics
Building 305, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-M.Sc.: ISSN XXXX-XXXX

Summary

The main focus of this master thesis is to solve an inverse problem using a stochastic method. The inverse problem is based on a diffraction problem from an experiment conducted in Grenoble, France. The goal is to understand the mathematics behind the physical experimen,t and solve the problem using a stochastic method. One of the important aspects of this thesis is to investigate, whether a deterministic solution of the problem can be used to optimize the solution found by a stochastic method.

To understand how the stochastic method deals with this physical problem a simulation study is conducted. First a simulation study on a simple mathematical model is performed and analyzed. The findings from this simple model can then be applied, when dealing with the more complex mathematical model. There are different aspects of the stochastic method, which we are able to exploit. In the simulation studies one of the main focus areas will be exploring the performance of the method based on different starting guesses. Also the type and amount of noise on data will be tested in accordance with the performance of the method. When the model is fully understood, a complex problem is solved and analyzed.

Throughout the thesis the results are presented in various ways to illustrate how the method actually works.

Resumé

Dette speciales primære fokus vil være at løse et inverst problem ved hjælp af en stokastisk metode. Det inverse problem er baseret på diffraktionsdata, der stammer fra et fysisk eksperiment udført i Grenoble, Frankring. Målet er at forstå matematikken bag eksperimentet og formulere a priori viden til løsningen af problemet med en stokastisk metode. En af de vigtigste aspekter i denne opgave er at undersøge, om en deterministisk løsning af problemet kan bistå til at optimere løsningen af problemet.

Et simuleringsstudium er udført for at forstå, hvordan en stokastisk metode løser et problem af denne type. Først er et studium udført på en forsimplet matematisk model og derefter analyseret. Konklusionerne derfra bliver udnyttet, når endnu et studium bliver udført med en mere kompleks matematisk model. Der er flere forskellige aspekter af metoden, som vil blive undersøgt og testet.

I simuleringsstudierne vil en af de gennemgående temaer være at undersøge metodens afhængighed af forskellige startgæt. Desuden vil der også blive fokuseret på mængden og typen af støj på data og hvilken rolle støjen har på løsningen. Når endelig modellen og metoden er gennemgået vil et komplekst problem blive løst og analyseret.

Gennem dette speciale vil resultaterne blive præsenteret på forskellig vis for at illustrere, hvordan metoden virker.

Preface

This master thesis was prepared atDTU Informatics, the Technical University of Denmark. It represents the final 30 ETCS point to fulfill the master program Mathematical Modelling and Computation. The work has been conducted during the autumn/winter of 2011 and 2012, more precisely the 1st of September 2011 to the 29th of February 2012. During this period the thesis has been processed under supervision of Professor Per Christian Hansen, DTU Informatics, Professor Klaus Mosegaard, DTU Informatics, and Senior Scientist Søren Schmidt, DTU Risø.

Lyngby, February 29th 2012

Ditte Iben Marcussen

List of Symbols

The following table lists the commonly used symbols in the thesis.

Symbol	Quantity	Dimension
$\boldsymbol{A}$	system matrix	$m \times n$
$\boldsymbol{a}_i$	i'th row of $\boldsymbol{A}$	m
$\boldsymbol{b}$	right-hand side	m
$\boldsymbol{b}^{\text{exact}}$	exact right-hand side	m
b_j	j'th element of the vector $\boldsymbol{b}$	scalar
$\boldsymbol{e}$	noise vector	m
e_j	j'th element of the vector $\boldsymbol{e}$	scalar
k	normalization constant	scalar
m,n	matrix dimensions	scalars
N_d	no. of gridpoints of the detectors	scalar
N_s	no. of gridpoints of the source in 4D problem	scalar
N_w	no. of gridpoints of the source in 2D problem	scalar
N_ϕ	no. of gridpoints of the radial grid in 4D problem	scalar
N_θ	no. of gridpoints of the angular grid	scalar
ϕ	radial angle	
$\mathcal{P}$	Poisson distribution	
σ_i	i'th singular value	scalar
Σ	matrix with singular values in the diagonal	$m \times n$

$\boldsymbol{U}$	matrix with all left singular vectors	$m \times m$
$\boldsymbol{u}_i$	i'th left singular vector	m
$\boldsymbol{V}$	matrix with all right singular vectors	$n \times n$
$\boldsymbol{v}_i$	i'th right singular vector	n
θ	angle of the ray	
$\boldsymbol{x}$	solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$	n
$\boldsymbol{x}^{\text{exact}}$	exact solution	n
z, w, y, t	cartesian axes in the set-up of the problem	
ρ	prior probability density function	
φ	posterior probability density function	
L	Likelihood function	
s	step length	scalar
c	rescaling constant	scalar
$\mathcal{E}$	expected value	
μ	standard deviation related to Gaussian distribution	scalar
S	misfit function	
$\hat{\theta}$	mean of location of θ	scalar
$\hat{\beta}$	standard deviation of θ	scalar

Contents

Chapter 1

Introduction

The physical experiment that underlies this entire thesis can in short terms be described as X-rays, which are sent through a small sample, diffracted and detected afterwards. The whole idea of conducting this experiment is to allocate the orientation of grains inside the crystal lattice from the detections of the rays penetrading the sample. The details of the experiment will be described in the next chapter.

The physical experiment plays an important role as to the motivation of this thesis. In [5] a slightly simpler problem than the one, we will deal with, is solved, but it might be possible to make use of another method, when solving it. This thesis will deal with a stochastic way of solving a more complex problem, and an attempt to combine a stochastic and deterministic part to a hybrid method.

First of all it is necessary to get a thorough understanding of the physical set-up to be able to deal with the mathematical model and also to translate the knowledge into prior information for the stochastic method. The prior information has to be in accordance with the actual experiment, the observations the scientists have done and also the physical limitations of the experiment and the results. Therefore the set-up of the stochastic method is based on cooperation between the realistic set-up and the actual implementation.

The model will be constructed as a General Ray Tracing Problem. When work-

ing with the model it reveals some weaknesses, which affects the performance of the solution method. That is of course a downside, but in the process of gaining an understanding on how the method works and the effect of combining a stochastic method with a deterministic one, it does not make a great difference.

It is important to always keep in mind throughout this thesis, that after this study of the method related to the physical experiment, we are not able to solve the full realistic problem. Instead we have obtained knowledge about a part of the problem, which can be utilized in later studies. The most essential knowledge to obtain in this thesis is the investigation of the role of the deterministic part in the stochastic solution method. After this thesis we shall be able to conclude, whether or not it is a good idea to combine the two methods or if one of them is sufficient in reaching a solution, which will satisfy the scientists.

In this thesis we will start off by describing the physical experiment in details along with the mathematical model based on the experiments. Then we will go through the relevant theory about discrete inverse problems and also stochastic methods. When describing the solution method we will discuss the advantages and disadvantages of using a stochastic method and verify the choices made in our method. As described in the summary we will perform simulation studies on a simplified problem and analyzing the results, before taking it a step further and analyze the results of tests on the full problem. At last a discussion takes the reader through all the important aspects of the results and method, and an outlook, which deals with all the possible directions, where the thesis could be extended if more time was available.

Chapter 2

Model Problem

As mentioned briefly in the introduction the overall idea with this thesis is to try to reconstruct orientations of the lattice within a small crystal of material by sending horizontal X-rays into the material and then detect the scattered rays emitted from the crystal. To solve this problem it is necessary to get a thorough understanding of the physical problem, both the precise set-up parameters and also the actual trace of the rays used. We start by describing the physics in the experiment and then make the necessary assumptions and modifications, so the set-up can be translated into a mathematical form, we are be able to solve using numerical calculations.

2.1 Underlying Physics

The experiments, which this thesis is based on, are done at a synchrotron facility in Genoble, France. The synchrotron facility ensures that it is possible to conduct experiments with high energy X-rays. The aim of the experiments is to investigate several properties including grain growth, grain orientation and neighboring relationships in a sample of metal using a method called three-dimensional X-ray diffraction (3DXRD) microscope - for details see [5]. The set-up describes high energy rays, which are sent through a sample and then the rays are scattered and collected in a charge-coupled device (CCD). The

beams are sent into a small layer in the sample at the time, and then the rays are scattered in parallel directions, due to the structure of the grains. The grains are what we call perfect grains, which means that the atoms or molecules are arranged in a symmetric structure and that affects the direction the rays are scattered in. Since the beams are scattered in parallel directions, one CCD is enough to detect the beams.

The next step is to look at beams scattered in a crystal with deformed structure - a so-called non-perfect crystal. The beams will deviate from the parallel directions with the perfect grains due to the structure wihin the crystal. Result of this will be that one detector is not enough to locate the origin of each ray in the sample. Therefore it is necessary to introduce more detector planes. The scattered beams are always emitted in a straight lines, so we need at least two detector planes to identify this line. In this modified set-up three detector planes are introduced, where the third plane is used to describe the angle distribution and also to verify the line described by the first two detectors. The detections from these three CCD's will represent the data used to reconstruct the structure within the crystal. In Figure 2.1 the set-up is sketched with both the sample and several detectors. In the figure the distances between the detectors do not reflect the true distances. In the real experiment the two first CCD's are placed very close to the source, whereas the third detector is placed further away. The first two detectors are therefore referred to as near-field detectors and the third as far-field detector.

Since this set-up is very complex to describe in a mathematical model, we will look at a subproblem of this, where we look at one of the pairs of detectors. If we get a thorough understanding about this part of the problem we might be able to extend our knowledge to the whole problem. This subproblem is illustrated in Figure 2.1, where we see the three detectors and the source. The plane just in front of the sample will be discussed in the next section.

2.2 Introducing the Fictitious Plane

Now we have introduced the underlying physics behind the general ray tracing problem. The only thing is, that the general rays tracing problem cannot reconstruct a three dimensional sample from the detections of three CCD's. Therefore the problem has to be simplified. The simplification consists of an imaginary source plane located just in front of the sample. The idea is that the beams emitting from the edge of the sample parallel to the source plane, we assume emit directly from the source plane instead. Then we only have to reconstruct a plane and not the whole crystal. The new set-up is seen in Figure 2.2. If we

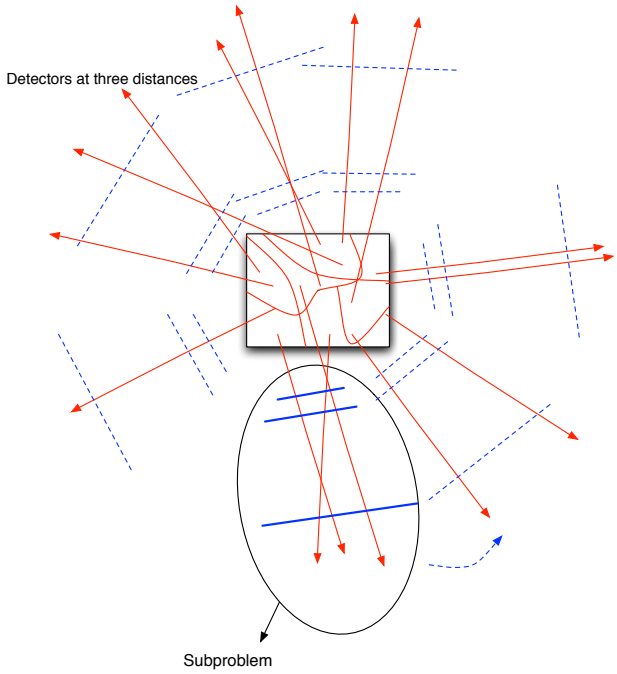

Figure 2.1: The experimental set-up.

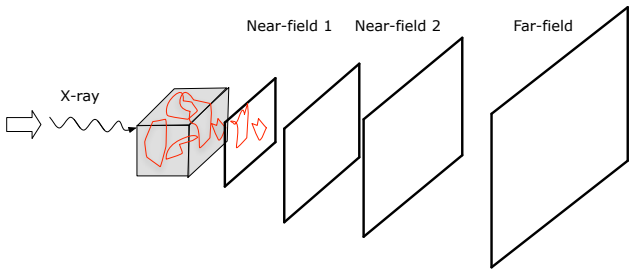

Figure 2.2: The experimental set-up of the subproblem. Reprinted with permission from [7]

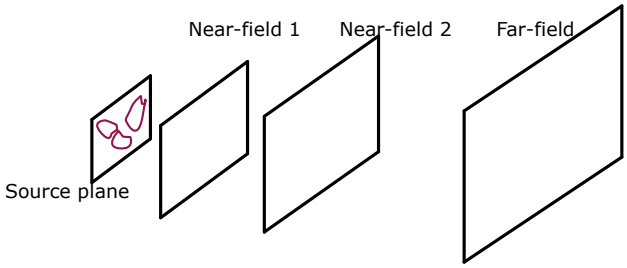

Figure 2.3: The simplified set-up with the imaginary source plane. Reprinted with permission from [7]

are able to solve this problem the same procedure can be used to reconstruct the whole sample based on measurements for different positions of the sample as illustrated in Figure 2.1. In this thesis the aim will be to reconstruct the intensity distribution at the source plane based on two spatial parameters describing the coordinates of the source plane, and two angular parameters which will describe the angles, where the rays are emitted.

Each ray is emitted in a cone shape, therefore two angular parameters are necessary. Reconstructing the source plane, based on the rays detected at the three detectors, corresponds to a four-dimensional problem. So the General Ray Tracing problem will produce four-dimensional projections, which can be used to describe a six-dimensional problem. The six-dimensional problem arise when the sample is rotated, so the projections can be used to reconstruct the whole sample as shown in Figure 2.1.

The three CCD's detect all the rays, which emit from the source plane and hit the CCD's. Each CCD is divided into a number of pixel, 2048 pixels in each direction, and the rays are detected pixelwise. This means that the detections by nature only have a discrete interpretation. The source plane contains a continuous distribution of rays. To be able to solve the problem, the source plane has to be discretized along with the angular coordinates. The number of pixel, we divide the spatial and angular coordinates into is limited by memory capacity, and therefore we have to choose a relative small amount of pixels. The exact number will vary dependent on the specific experiment, and will be mentioned in the sections dealing with the results of the reconstructions. What will be fixed through the whole thesis is the distances between the source plane and the detectors. The first CCD is placed 8 mm from the source, the second CCD 18 mm and the third CCD will be placed 500 mm from the source. After introducing this source plane just in front of the crystal we are now able to describe and discretize the problem, so we are able to solve it using a sampling

method.

Chapter 3

Mathematical Model

To solve the problem numerically it is a necessity to discretize the problem. In this section we will describe the continuous model along with the discretized one and discuss the relevant aspects of noise in the problem setting.

3.1 Continuous Model

To formulate a mathematical model we have to make a detailed description on how the rays are emitted and detected. Light is emitted from each point at the source in a cone shape and then detected at several pixels at each detector plane. The size of the radius in the circle at the end of the cone depends on the distance between the source and the current detector plane. We assume the detector planes are located, such that a straight line from orego of the source with an angle equal to zero hits each detector plane in their orego as well. This ensures the detector planes are not shifted. We also assume that the rays do not loose intensity as they pass through the detectors. In this model we assume that there is no blurring occurring at the CCD's, when they detect the rays. In Figure 3.1 the set-up behind the mathematical model is illustrated.

The intensity distribution function denoted f is dependent on four different variables and lives at the source plane. The four variables are two spatial coor-

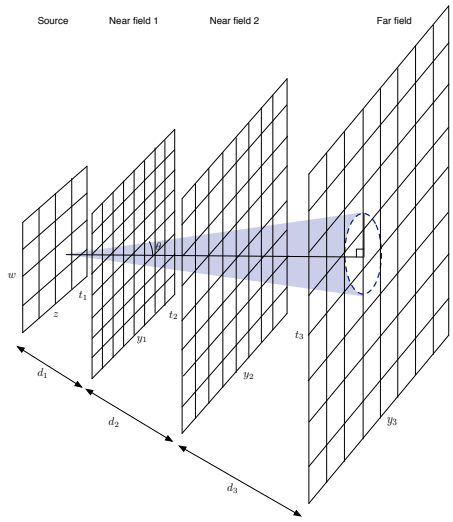

Figure 3.1: Illustration of the problem set-up with two spatial coordinates. Reprinted with permission from [7]

dinates (z, w), which describes the point where the light is emitted from and a set of angles (ϕ, θ), where θ describes the angle between the plane and the light emitted, and ϕ the circle at the detectors. The variables are continuous and lie within the intervals

$$w, z \in [-0.5, 0.5], \quad \phi \in [0, 2\pi], \quad \theta \in [0, \pi/2]. \tag{3.1}$$

To set up the model we look at one specific pixel on a specific detector k at a certain time. The detection g corresponds to the rays coming from all points on the source and detected at a specific pixel on a specific detector. This is illustrated in Figure 3.2. The coordinates of the detectors are denoted (y^k, t^k). Looking at just one ray emitted from one point at the source, and detected at one pixel on detector k, the detection can be described by

$$\Delta g_k(w_i, z_j, y_{k_l}, t_{k_m}) = \int_{\theta_1}^{\theta_2} \int_{\phi_1}^{\phi_2} f(w_i, z_j, \phi, \theta) d\phi d\theta. \tag{3.2}$$

The parameters $\theta_1, \theta_2, \phi_1, \phi_2$ correspond to the boundaries of integration for a specific pixel on a detector plane. To find the total contribution of rays detected on each pixel on detector k we add up all the rays emitted from each point on the source.

$$g_k(y_{k_l}, t_{k_m}) = \int_{w_1}^{w_2} \int_{z_1}^{z_2} \Delta g_k(w, z, y_{k_l}, t_{k_m}) dz dw \tag{3.3}$$

The parameters w_1, w_2, z_1, z_2 correspond to the boundaries of integration for a specific point on the source plane. This formulation of the continuous model will be the basis of the derivation of the corresponding discrete model introduced in next section.

3.2 Discrete Model

To discretize the problem such that a system of linear equations of the form $\boldsymbol{Ax} = \boldsymbol{b}$ is reached, the continuous model above will be used. First of all we have to introduce a set of sub-pixels $p \times q$, which will represent each pixel at each detector. The reason for this is that in each pixel on the detector, rays hit with different values of θ and ϕ. To separate these values, the pixels have to be divided into sub-pixels - for further details see [7].

The intensity distribution function will represent the $\boldsymbol{x}$ in the system, and therefore we start by discretizing this. The domains of the four variables have to be discretized. The spatial coordinates are assumed to have the same grid resolution, N_s, in both directions, and the angular variables will have N_θ and N_ϕ grid points respectively. The vector $\boldsymbol{x}$ will represent the intensity distribution function $f(w_i, z_i, \phi_m, \theta_n)$, where $i, j = 1, \ldots, N_s$, $m = 1, \ldots, N_\phi$ and $n = 1, \ldots, N_\theta$. Adding this up will result in a $\boldsymbol{x}$ with number of element equal to $N_s^2 \cdot N_\theta \cdot N_\phi$.

The data $\boldsymbol{b}$ consist of all the detections on each detector plane k. Each detector has $N_d \times N_d$ pixels, so the total number of detections for the three detectors add up to a number of $3 \cdot N_d^2$. Hence the right-hand side $\boldsymbol{b}$ will consist of $3 \cdot N_d^2$ elements stacked as a vector. In the end this means that the system matrix $\boldsymbol{A}$ will have the dimensions $3 \cdot N_d^2 \times (N_s^2 \cdot N_\theta \cdot N_\phi)$. Letting $\boldsymbol{F}$ be a four-dimensional matrix of dimensions $N_s \times N_s \times N_\phi \times N_\theta$ representing the discrete

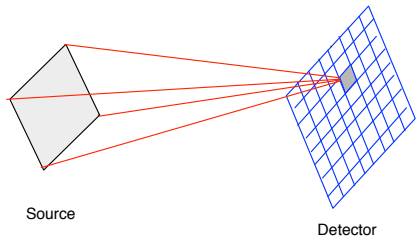

Figure 3.2: Illustration of the light emitted from the source and detected at one pixel on one detector.

intensity distribution at the source and letting $\boldsymbol{G}$ be a two-dimensional matrix representing the discrete detections at the detectors. Then equation (3.2) in discrete terms will be expressed as

$$\Delta \boldsymbol{G}_k(w_i, z_j, y_{k_l}, t_{k_m}) = h_\phi \cdot h_\theta \sum_{r=1}^{q} \sum_{s=1}^{p} \boldsymbol{F}(w_i, z_j, \phi_r, \theta_s). \tag{3.4}$$

Here h_θ and h_ϕ represent the grid spacing in the domains of the θ and ϕ. As in the continuous case the total detection at each pixel for each detector is expressed as a sum of (3.4) for all pixels at the source

$$\boldsymbol{G}_k(y_{k_l}, t_{k_m}) = \sum_{i=1}^{N_z} \sum_{j=1}^{N_w} \Delta \boldsymbol{G}_k(w_i, z_j, y_{k_l}, t_{k_m}) \tag{3.5}$$

$$= \sum_{i=1}^{N_z} \sum_{j=1}^{N_w} h_\phi \cdot h_\theta \sum_{r=1}^{q} \sum_{s=1}^{p} \boldsymbol{F}(w_i, z_j, \phi_r, \theta_s). \tag{3.6}$$

Now we stack the columns of the intensity function $\boldsymbol{F}$, so

$$\boldsymbol{x} = vec(\boldsymbol{F}). \tag{3.7}$$

In the same way the right-hand side is formed by all the detections $\boldsymbol{G}$ stacked in a vector. Since the right-hand side consists of three detectors and therefore three detection matrices $\boldsymbol{G}_k$, these have to be stacked in terms

$$\boldsymbol{b}_k = vec(\boldsymbol{G}_k). \tag{3.8}$$

Then in the end the three vectors $\boldsymbol{b}_k$ will be stacked to result in one right-hand side $\boldsymbol{b}$ of dimensions $3 \cdot N_d^2 \times 1$. The model matrix $\boldsymbol{A}$ is constructed by running through all unit vectors and then the detections will become the columns of $\boldsymbol{A}$. So in the end we reach a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, which we are able to solve using a sampling method.

One important aspect of the discretization is of course the errors that occur due to discretization. These errors are related to grid spacing, the finer the grid the smaller discretization errors. As mentioned earlier the dimensions of the detectors are 2048×2048 in the physical set-up, but we do not need to work with full scale dimensions to understand the problem. Working with the full scale resolution would imply long computation times and large use of memory - this will be illustrated in Section 6.5. This discrete model introduced contains two spatial dimensions. Later in this thesis we will deal with a simplified model with one spatial coordinat. The model related to the simpler problem will be described in Chapter 6.

3.3 Noise

When working with simulated data it might be tempting to do the introductory calculations with noise free data. But in this thesis where we want to solve the inverse problem using a sampling method, working in a noise free environment is not a good idea and not realistic. The details about why it is not a good idea will be discussed further in Chapter 5. For all types of noise we can assume that there exist an exact solution $\boldsymbol{x}^{\text{exact}}$. Then there is a corresponding right-hand side $\boldsymbol{b}^{\text{exact}} = \boldsymbol{A}\boldsymbol{x}^{\text{exact}}$, which will be noise free. Then the model for the right-hand side will be

$$\boldsymbol{b} = \boldsymbol{b}^{\text{exact}} + \boldsymbol{e}, \quad \text{where} \quad \boldsymbol{b}^{\text{exact}} = \boldsymbol{A}\boldsymbol{x}^{\text{exact}}, \tag{3.9}$$

where $\boldsymbol{e} \in \mathbb{R}^m$ will be a vector representing the noise in the data. We will work with two types of noise in this thesis and they will now be introduced.

3.3.1 Gaussian Noise

Gaussian noise also referred to as Gaussian White Noise is based on the error term $\boldsymbol{e}$ described by a Gaussian distribution with zero mean and standard deviation μ. The expected value of the noise is defined as

$$\mathcal{E}(\boldsymbol{e}) = 0, \quad \mathcal{E}(\|\boldsymbol{e}\|_2^2) = \mu^2 m. \tag{3.10}$$

This only holds if the elements in $\boldsymbol{e} \in \mathbb{R}^m$ come from the same Gaussian distribution with zero mean and standard deviation μ.

One important aspect of data containing noise is that if $\boldsymbol{A}$ is ill-conditioned then the naive solution $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ is very sensitive to errors on data - see [1] for further explanation.

3.3.2 Poisson Noise

When modelling real world problems, noise is an important aspect. Since the data is based on a real physical experiment, there will always be noise on data. With the physical set-up in mind a realistic type of noise on the simulations could be Poisson noise. Poisson noise is often used, when a finite number of particles carrying energy, in this case rays, are detected as intensity. Opposite from Gaussian white noise, Poisson noise only depends on one parameter. If data

follows a Poisson distribution the i'th component of the data can be written as

$$b_i \sim \mathcal{P}(b_i^{\text{exact}}), \quad i = 1, \ldots, m. \tag{3.11}$$

The mean and variance will then be $\mathcal{E}(b_i) = b_i^{\text{exact}}$ and $\mathcal{V}(b_i) = b_i^{\text{exact}}$ respectively. When the Poisson noise is represented as above, it is hard to control the proportionality factor between data and noise. From the experiments it is known that the relative noise level is approximately 10 %. The relative noise level (RNL) is defined as the norm of the error divided by the norm of the exact vector. Then the RNL for data will be expressed as

$$RNL = \frac{\|b - b^{\text{exact}}\|_2}{\|b^{\text{exact}}\|_2}. \tag{3.12}$$

When we are specifying a certain level on Poisson noise, it is not as simple to modify this level as in for instance Gaussian noise. The RNL is dependent on the level of intensity in the exact solution. Therefore to change the RNL, it is necessary to multiply the solution with a constant c. So rescaling the data $\hat{\boldsymbol{b}}^{\text{exact}} = c\boldsymbol{b}^{\text{exact}}$ will be the first step in finding the RNL, which corresponds to the RNL in the experiments. Looking at the norm, the same constant is used $\|\hat{\boldsymbol{b}}^{\text{exact}}\|_2 = c\|\boldsymbol{b}^{\text{exact}}\|_2$. This result will help us later in finding the constant c. We know that in this set-up the expected value of the error level can be expressed as

$$\mathcal{E}(\|\boldsymbol{e}\|_2^2) = \|\boldsymbol{b}^{\text{exact}}\|_1. \tag{3.13}$$

See [7] for further details. We are now able to find the expected value for RNL as

$$\mathcal{E}(RNL) = \frac{\mathcal{E}(\|\boldsymbol{e}\|_2)}{\mathcal{E}(\|\boldsymbol{b}^{\text{exact}}\|_2)} \backsimeq \frac{\|\boldsymbol{b}^{\text{exact}}\|_1^{1/2}}{\|\boldsymbol{b}^{\text{exact}}\|_2}. \tag{3.14}$$

Likewise we can find the expected value for $\widehat{RNL}$

$$\mathcal{E}(\widehat{RNL}) = \frac{\mathcal{E}(\|\hat{\boldsymbol{e}}\|_2)}{\mathcal{E}(\|\hat{\boldsymbol{b}}^{\text{exact}}\|_2)} \backsimeq \frac{\sqrt{c}\|\boldsymbol{b}^{\text{exact}}\|_1^{1/2}}{c\|\boldsymbol{b}^{\text{exact}}\|_2} = \frac{1}{\sqrt{c}}\mathcal{E}(RNL). \tag{3.15}$$

Using this we can find the constant c, so the RNL will be approximately 10 %. So for instance if we want to reduce RNL with a factor 2, we have to multiply our data with 2^2.

Chapter 4

Discrete Inverse Problem

In this chapter the underlying theory used throughout this thesis will be described. There will be a short introduction to the concept of inverse problems and different tools used in relation to.

4.1 Inverse Problems

An inverse problem covers the set-up, where some external knowledge is known - though often in a noisy version and the aim is to recover some internal or "hidden" data. The phenomenon arises in different fields of science, to mention a few - medical, geophysics and astronomy. A well-known example of an inverse problem can be reconstructing a sharp image from a blurred version. This problem is not trivial to solve, due the ill-posedness of the problem. This will be described in further details in this section. Inverse problems can take both a continuous and a discrete form. The continuous form can be expressed as a Fredholm integral equation of first kind. Discrete inverse problem will be the formulation we will be working with in this thesis. It is necessary to continue with a discrete formulation of the problem, otherwise it is not possible to solve using computer science. The discrete inverse problem is represented by the linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^m$. The vector $\boldsymbol{b}$ represents the right-hand side, which is the well-defined data often

containing noise. The matrix $\boldsymbol{A}$ is called the system matrix and it describes the relationship between known data and the unknown factor. The unknown factor is of course the vector $\boldsymbol{x}$, which we wish to approximate. For convenience we will often use the formulation, where $\boldsymbol{A}$ is square, so $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$. If $m > n$ the system is overdetermined and the problem will be a least squares problem defined as $\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^m$. If $m < n$ the system is underdetermined.

As mentioned above an inverse problem is categorized as an ill-posed problem. This is concluded from the definition of well-posed linear problems stated by Hadamard - see [1]. Hadamard's definition says that a problem is well-posed if the following three statements is fulfilled. Existence; the problem *must* have a solution, uniqueness; there must be only *one* solution and stability; the solution must depend *continuously* on the data. If one or more statements is violated, the problem is ill-posed. If a problem is ill-posed, it is still solvable. The discrete Picard condition defines, when the problem is solvable. This will be described later in this chapter.

4.2 Singular Value Decomposition

One of the tools, which will be used to reconstruct test images using Monte Carlo method is the Singular Value Decomposition - hereafter denoted SVD. This tool can also be used to analyze the existence and the instability of the solution. The SVD can be helpful for all finite-dimensional matrices and for a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ it is defined as

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{n} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T, \tag{4.1}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the singular values $\sigma_1, \ldots, \sigma_n$ in the diagonal. These values is non-increasing, so $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$. The matrix $\boldsymbol{U} \in \mathbb{R}^{m\times n}$ contains the left singular vectors as columns, so $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n)$, and similarly $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ contains the right singular vectors as columns, so $\boldsymbol{V} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$. These matrices have orthonormal columns, so the following holds

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}. \tag{4.2}$$

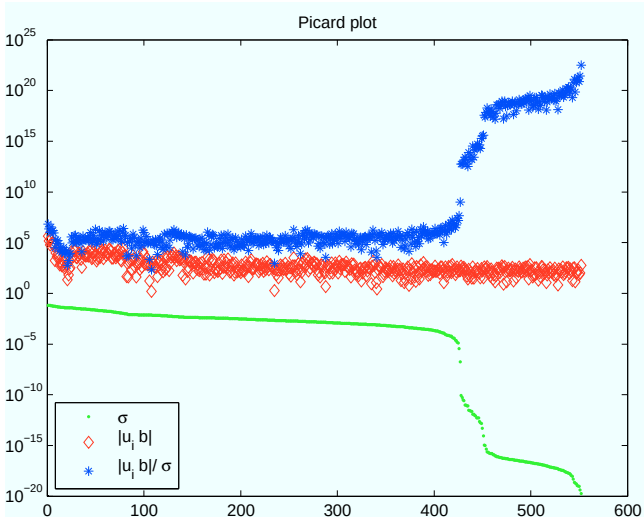

Figure 4.1: Picard plot representing the singular values and the SVD coefficients of the matrix $\boldsymbol{A}$.

4.2.1 Discrete Picard Condition

Now the SVD is introduced, but what can it be used for? It can be used to find the solution of the inverse problem $\boldsymbol{Ax} = \boldsymbol{b}$. Looking at the formulation of the naive solution for the case where $\boldsymbol{A}$ is square - derived in [1]

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} = \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i, \tag{4.3}$$

we notice two important aspects. One, that the right singular vectors $\boldsymbol{v}_i$ seem to have great impact on the solution. Two, that the fraction $\frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i}$ is increasing due to the descending singular values. So what happens when the singular values level off due to rounding errors? The naive solution is dominated by noise, so therefore the Discrete Picard Condition is introduced. If τ is defined as the level, where the singular values σ_i is leveled off due to rounding errors, the Discrete Picard condition is satisfied as long as the coefficients $|\boldsymbol{u}_i^T \boldsymbol{b}|$ decay faster than the singular values σ_i for the values larger than τ. This condition is often verified by a so-called Picard plot, where the singular values are plotted along with the SVD coefficients and the relationship between those two $|\boldsymbol{u}_i^T \boldsymbol{b}|/\sigma_i$. If the values of this fraction is starting to increase the Picard Condition is not satisfied any more. A Picard plot is seen in Figure 4.1 and for further details see [4]. Often the Picard plot is used to find the value τ, and then discard the rest of the SVD

coefficients after this value. Then the optimal solution is found in stead of the naive solution. This is called the truncated SVD.

The theory behind the problem and some of its properties has now been discussed. SVD and the Discrete Picard Condition have mainly been described to illustrate how the difficult solving an inverse problem is. The tools will not be used in the analysis of the results in the thesis in Chapter 6 and 7, but they are used in the introductory investigations in Appendix A. The introductory investigations were conducted in order to get a basic understanding about the problem and how the stochastic method behaved, when solving the problem.

Chapter 5

Sampling Methods

The main focus of this thesis will be on a stochastic method, which differs from a deterministic method both in formulation and also the results are represented in another way. The reason why the deterministic method is described is due to the aim of using a hybrid of the two methods to solve an inverse problem. It will be interesting to see if taking the best of the two worlds will result in a better solution than using just one of them. Sampling methods is a method to statistically choose a subset of individuals (a sample) from a larger set (a population) to describe characteristics of the whole population. The term sampling methods cover many different methods, where the *Simple Random Sampling* is a simple and widely used method. The idea is that each individual sample is chosen randomly and has the same probability to be chosen through the whole sampling process. Within this method we find the Monte Carlo method, which belong to the class of sampling methods. In this chapter we will present a sampling method in general terms and then describe how the method is implemented.

5.1 Monte Carlo Method

The reason why the Monte Carlo method belong to the class of sampling method is that the method randomly generates solutions from a probability distribution

to simulate the process of sampling from an actual population. This probability distribution has to be chosen wisely, since it has to correspond to the data we have. This choice will be discussed later in this chapter and also in Chapter 6. The field of Monte Carlo methods covers different variations methods, to mention a few: Simulated annealing, Genetic algorithms and Neighborhood algorithm. In this section the mathematics behind the Monte Carlo method will be described and the different variations of the method will be looked into.

We already introduced inverse problems in the deterministic form,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{5.1}$$

where $\boldsymbol{b}$ is the observed data and $\boldsymbol{x}$ is the model parameters, which are not observable. The inverse problem will be formulated in a stochastic form as the relationship between model parameters and data and prior information. We will work with a priori information in terms of a probability density function $\rho_x(\boldsymbol{x})$ and an a posteriori information described as a probability density function $\varphi(\boldsymbol{x})$. The relationship between these two densities is expressed as

$$\varphi(\boldsymbol{x}) = k\rho_x(\boldsymbol{x})L(\boldsymbol{x}), \tag{5.2}$$

where k is a normalization constant and $L(\boldsymbol{x})$ is a likelihood function, which describes the degree of fit between observed data $\boldsymbol{b}$ and predicted data using the model $\boldsymbol{x}$. It is defined as

$$L(\boldsymbol{x}) = \rho_b(\boldsymbol{A}\boldsymbol{x}). \tag{5.3}$$

Here ρ_b is the prior probability density function describing the data. Alternatively the misfit function $S(\boldsymbol{x})$ can be used instead of the likelihood function. Using the misfit function the problem becomes an optimization problem, and then the exponential part of the likelihood function is avoided. The relationship between these two functions is described as

$$L(\boldsymbol{x}) = k\exp(-S(\boldsymbol{x})). \tag{5.4}$$

The baysian approach to this inverse problem is to describe the a posteriori density, which contains all the information we have. The expression (5.2) refers to the probability distribution that describes the solution to the inverse problem.

5.1.1 Probability Density

Now we know that the probability density describing the solution consists of the prior probability function and the likelihood function. The likelihood function often depends on the noise added to data, and therefore it is very problem

dependent. It will described later in this chapter, what likelihood function we will use. The prior probability density describes the prior knowledge that we have about the solution. It can be defined in two different ways. The first is an explicit formula, which is often not available. The other method is by defining a random process, whose output is different models all representing pseudo random realization of the distribution $\rho_x(\boldsymbol{x})$. The latter will be explained in further details later in this chapter. Sampling the a posteriori probability density is more complex. One method of sampling the density is the extended Metropolis algorithm:

Extended Metropolis Algorithm Given a random function $V(\boldsymbol{x})$, which samples the prior probability density $\rho_x(\boldsymbol{x})$ if applied iteratively:

$$\boldsymbol{x}^{(n+1)} = V(\boldsymbol{x}^{(n)}), \tag{5.5}$$

and a random function $U(0,1)$, which generates numbers in the interval [0,1] and lastly a random function W, which iteratively generates the next parameter vector $\boldsymbol{x}^{(n+1)}$ from the current parameter vector $\boldsymbol{x}^{(n)}$ altogether gives the algorithm:

$$\boldsymbol{x}^{(n+1)} = W(\boldsymbol{x}^{(n)}) = \begin{cases} V(\boldsymbol{x}^{(n)}) & \text{if} \quad U(0,1) \leq \min\left[1, \frac{L(V(\boldsymbol{x}^{(n)}))}{L(\boldsymbol{m}^{(n)})}\right] \\ \boldsymbol{x}^{(n)} & \text{otherwise} \end{cases} \tag{5.6}$$

which asymptotically samples the posterior probability density $\varphi(\boldsymbol{x}) = kL(\boldsymbol{x})\rho(\boldsymbol{x})$, where k is a normalization constant.

This extended version only works if V is irreducible and aperiodic. For furhter details see [3]. The output of this method is then a probability density describing the solution. Compared to a deterministic method where the solution is the same no matter how many times you run the calculation, this method will end up with slightly different densities, which all fits the observable data. Therefore the analysis of the results will also differ from the deterministic one. Instead of one solution the Monte Carlo method produce a finite large number of solutions. How to visualize these solutions will be a main topic in Chapters 6 and 7. Now the basic theory about the method is introduced, now it is time to get a closer look at the different modules in the method.

5.1.2 Modules

The method can be divided into several boxes or modules. That helps to get an overview over the method.

Starting Guess To start the method it needs a guess on the solution. This starting guess has influence on how the method performs, and therefore is this module essential. Using simulated data it is possible to use $\boldsymbol{x}_{\text{exact}}$ as starting guess. But of course this is not realistic, so alternatively a possible sample from the prior distribution, or choosing a solution consisting of zeros could be an option. Another idea is also to obtain a deterministic solution to the problem and use that as starting guess.

Realization One of the most important modules is the module, where all the realizations are made. A realization is a random generated guess on the solution, which is sampled from the prior probability density. The realization might be discarded, so it is not the same as a solution.

Misfit/Likelihood Function From each realization a value is calculated, either using the misfit or the likelihood function. The value is calculated based on the residual of the current model.

Accept Criteria Each realization is sent through the Accept Criteria module. Here it is decided based partly on a random process partly on the value calculated above, whether this realization is accepted to represent the solution or it is rejected, meaning that it did not fit the a posteriori probability density.

The implementation of the modules will be described in Section 5.3.1

5.2 Properties

Some of the properties of the Monte Carlo method have to be described into detail to understand how to evaluate the method and the corresponding solutions. As mentioned in the first module the starting guess has great impact on the behavior of the method. This can be verified by looking at a so-called misfit plot. A misfit plot is all the values calculated in the misfit function corresponding to accepted realizations shown as a function of the iteration numbers. In Figure 5.2 two misfit curves are shown representing the use of two different starting guesses. We can conclude that when using the exact solution as starting guess the misfit values level off after relative few iterations. Opposite, when using a starting guess far from the exact solution the method needs many iterations before the misfit values level off. The solutions corresponding to the iterations before the values are leveled off should not be included when describing the model, since these solutions is not on the right level yet. These iterations are called the burn-in period - see [6]. This burn-in period tells us something about how many iteration it takes to get to the solutions describing the model. What

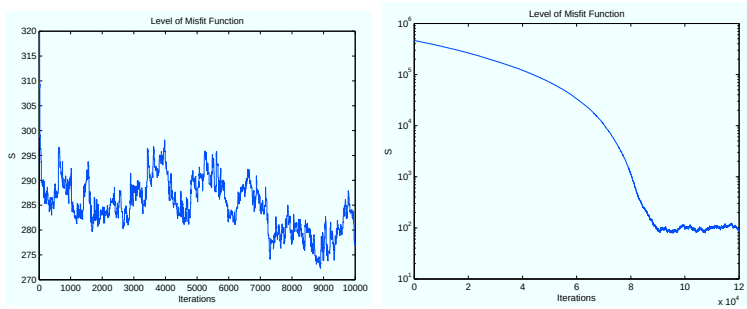

Figure 5.1: Two misfit plot corresponding to using $\boldsymbol{x}_{\text{exact}}$ (left figure) and zeros (right figure) as starting guess.

we also see from Figure 5.2 is that the value of the level, where the misfit values stagnate, varies for different problems. This value is dependent on the number of solution variables and also on the number of degrees of freedom in the problem.

5.2.1 Pros and Cons

In this section we will discuss some of the advantages and disadvantages by using a sampling method to solve an inverse problem. It is an important discussion, since we want to construct a hybrid of the deterministic and stochastic methods, and therefore it is important to take the best from the two worlds. We want to focus on the advantages of using a stochastic method and one of the most important ones is that the method is suitable for large-scale problems. In a deterministic method it is impossible to avoid the use of the matrix $\boldsymbol{A}$. This matrix grows rapidly, when the problem dimensions increase. Therefore deterministic solvers have a hard time dealing with large-scale problems. In a stochastic method avoiding matrix multiplication is actually possible. In this Monte Carlo method only a forward operation is necessary, and that can be done without using the model matrix. This will be described in detail in 6.5.

One of the biggest differences between a deterministic and a stochastic method is the way the solutions are processed and visualized. The stochastic method find a huge number of possible solutions all fitting the observed data. These solutions are not the optimal solution as the deterministic method finds. This is illustrated in Figure 5.2. Using a deterministic solver the solution converges toward the exact solution compared to using a sampling method, where many solutions are computed and they converge to solutions close to the exact solution. The solutions located in a cloud around the exact solution correspond to the

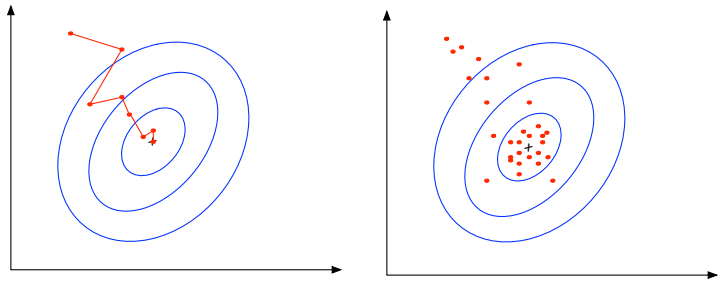

Figure 5.2: In the left figure some iterations are simulated corresponding to for instance a CGLS solver and the right figure correspond to samplings of a sampling method.

solutions after the burn-in period. The distance between these solutions and the exact solution is dependent on the amount of noise on data. The more noise in data the greater distance.

Another advantage of using a stochastic solver is the ability to utilize a priori aspects. In physical experiment the scientists often have some prior knowledge of some of the parameters and have an idea of how the solution might look like. That kind of knowledge can be implemented in the method, so the realizations fit the prior restrictions that you have specified. It is possible to implement many types of restrictions.

You could see it as an disadvantage that the stochastic method does not converge to the exact solution if data does not contain noise, but on the other hand you have several solutions describing the errors within the solution. Again it is important to mention that it does not make any sense to use noise free data in this Monte Carlo solver. When a large amount of noise is added on data it might be impossible to solve the inverse problem using a deterministic method - see Chapter 4, and the method increases its performance as the level of noise decreases. Using a stochastic method it works in the opposite way. The smaller amount of noise the harder it is to find a density distribution describing the solution. If more noise is added the distribution becomes smoother and easier to find using the method. Of course putting too much noise on the data will imply almost no restrictions on the solution and therefore the accepted realizations can deviate a lot from the exact solution.

The ideal way to visualize the solutions is to show a video representing the all solutions, but since this is not possible in a report, we must visualize in a different way. One simple method is to find a mean solution of all the solutions

after the burn-in period and show this as one solution. Then it is possible to calculate relative errors and visualize the solution in a two-dimensional image.

5.3 Our Method

Now the Monte Carlo method is described in general terms, but when it comes to the actual implementation some choices have to be made. First of all, what algorithm should handle the rejection. We have chosen to use the Extended Metropolis algorithm, which is fairly simple.

Looking at the formulation of the likelihood function based on gaussian uncertainties on the data described by a covariance matrix $\boldsymbol{C}$

$$L(\boldsymbol{x}) = k \exp\left[-\frac{1}{2}(\boldsymbol{Ax} - \boldsymbol{b})^T \boldsymbol{C}^{-1}(\boldsymbol{Ax} - \boldsymbol{b})\right], \tag{5.7}$$

where k is a normalization constant, we want to avoid working with the exponential function. We choose to calculate the degree of fit between data and the realization by using the misfit function instead of the likelihood function to avoid the exponential function. We assume that the uncertainties are independent and identically distributed, therefore using equation (5.4) we get a misfit function of form

$$S(\boldsymbol{x}) = \frac{1}{2}\frac{\|\boldsymbol{Ax} - \boldsymbol{b}\|^2}{\sigma^2}. \tag{5.8}$$

Since the $\boldsymbol{b}$ is containing noise in the same magnitude as σ, the value of the misfit function will stay close to $\frac{f}{2}$, where f is the degrees of freedom.

When deciding what starting guess to use it gets a bit complex. Since we are working with simulated data, it would be a good idea to start by using $\boldsymbol{x}_{\text{exact}}$ as starting guess. That would give a good indication, whether the method works or not. On the other hand with real world data you will never have the exact solution, so it would not be an option. Therefore we choose to use $\boldsymbol{x}_{\text{exact}}$ as starting guess to test the method and then afterwards use several other starting guesses including a zero vector, the exact solution added noise and then a deterministic solution to the problem. In this way we test the method and also the robustness of it. The idea behind the latter is that the deterministic solvers find a solution close to the exact within a few iterations and using that it might decrease the burn-in period, resulting in fewer Monte Carlo iterations.

In the module where the realizations are produced the priori information decides how they should be constructed. Therefore we actually make a choice how the a

priori information is represented. This is based on the knowledge we have on the actual physical experiment. For instance we know that it is more likely that in some of the angles θ no rays are emitted. That can be taken into account, when we model the a priori probability density function. The more restrictions we have on our solutions the closer the realization will be at the exact solution. On the other the hand Monte Carlo methods prefer as few restrictions as possible.

5.3.1 matlab Implemtentation

Starting with the most simple function, the misfit function. It is based on the expression in equation (5.8) and in the code it is implemented as seen below

```
function S = misfit(b,x,A,sigma)
diff = A*x-b;
S = 0.5*norm(diff)^2/sigma^2;
```

σ represents the noise in data. The function where the realizations are made is varying dependenting on the problem we look at. The most simple implementation is using a Gaussian prior, which perturbates the current solution. This is implemented below

```
function x = realization(xcurr,s)
x = zeros(size(xcurr));
% Using simple Gaussian a priori
for ii = 1:length(xcurr)
x(ii) = xcurr(ii) + (2*rand-1)*s;
end
```

Using the prior related to the restricted θ values is more complex. Basically the implementation is a detailed description of the prior information and based on hard constrains.

Now we have the basic functions described and the algorithm itself needs to be implemented. The Extended Metropolis Algorithm described in Section 5.1.1 is implemented as below.

```
k = 0;
Scurr(1) = misfit(b,x0,A,sigma);
xcuur = x0;
for ii = 2:Nit
    xtest = realization(xcurr,s);
    Scurr(ii) = misfit(b,xcurr,A,sigma);
```

```
    Stest = misfit(b,xtest,A,sigma);

    % The next value is set to the previous value
    Scurr(ii) = Scurr(ii-1);

    % Acceptance
    if Stest <= Scurr(ii-1)
        xcurr = xtest;
        Scurr(ii)=Stest;
    else
        Pa = exp(-(Stest-Scurr(ii-1)));
        prob = rand;
        if prob <= Pa
            xcurr = xtest;
            Scurr(ii)=Stest;
        end
    end
    X(:,ii) = xcurr;
end
```

The implementation above is a simple version, where the a priori information is not an input to the function `realization`. Here the prior information is chosen to be the simple Gaussian prior. The more complex prior will be described, while dealing with different test problems in Chapter 6 and 7.

5.4 Noise within the Method

The sampling method takes noisy data as input and use it to calculate the residual for each realization constructed within the method. As we saw in the function above, where the misfit value is calculated, a parameter representing the noise level in data is used. So if the noise level in data is high, the misfit value value decreases and the chance of accept increases. Therefore the sampling method performs better this high levels of noise than the deterministic methods do. When testing it is possible to give the method noise-free data, but it still needs a parameter indicating how much noise there is on data. If the parameter is set to zero, the algorithm will not function. Therefore it is possible to set the parameter different from the actual noise on data. This will basicly means that the method thinks there is noise in data even though the data might is noise-free. The sampling method will not as discussed earlier find solutions satisfying the noise-free data, but solutions with corresponding data lying a distance between the exact data specified by the noise level parameter used in the misfit function.

Therefore we will always make sure that the parameter used in the misfit function correspond to the actual noise in data. In Chapter 6 and 7 test problems

are solved with different levels of noise, and we look at the consequences of the noise levels.

Chapter 6

Two-dimensional Problem

Now all the theory behind the method and the discretization of the mathematical model are described, and we can start solving the problem. Although it might seem as the next step in the process, we will start by taking a step back and focus on a simplified model. We simplify the mathematical model, so we only work with one spatial coordinate and one angle instead of two. Based on the aim of this thesis, which is to get a thorough understanding of the problem and how it is solved used a sampling method, it makes sense to start simple. Then we acquire some important knowledge, that can be used to solve the full four-dimensional problem afterwards. This section will only deal with this simplified model, and we start off by introducing it.

6.1 Simplified Model

The idea behind the simplified model is of course the same as in the full problem and this idea is seen in Figure 6.1. One important aspect that we have to deal with first is the range of the detectors and thereby the angle interval, which covers the three different planes in the set-up. If a ray emitted from the source is sent out with an angle, and it hits the edge of the first detector, it will not hit the edges of the two other detectors. Therefore it is necessary to modify the range of the detectors in order to obtain the maximum angle, the rays can have,

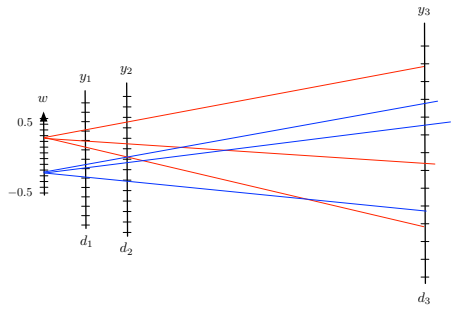

Figure 6.1: Problem set-up with one spacial dimension.Reprinted with permission from [7].

when they emit from the source. The original values are seen in Table 6.1 along with the modified ones, that will be used throughout this chapter and Chapter 7, when the simulation studies are performed.

Parameters for both laboratory and simulated set-up				
		d_1	d_2	d_3
	Distance	8 mm	18 mm	500 mm
Laboratory	Range θ_{max}	±1.54 mm ±0.25	±4.61 mm ±0.28	±51.2 mm ±0.10
Simulation	Range Range θ_{max}	±1.77 mm ±0.28	±4.61 mm ±0.28	±141.39 mm ±0.28

Table 6.1: Table of parameters for the laboratory set-up and the simulated set-up.

After dealing with this issue we can now begin formulating the two-dimensional model. Starting with the continuous case the intensity distribution function is again denoted f and is dependent on the two variables $w \in [-0.5, 0.5]$ and $\theta \in]-\pi/2, \pi/2[$. For the specific spatial coordinate w rays are lead out in different angles θ. This rays are detected at detector k at the j'th pixel. The total light intensity detected at this pixel is given by

$$g_k(y_j) = \sum_i \int_{\theta_{\text{start}}}^{\theta_{\text{end}}} f(w_i, \theta) d\theta, \tag{6.1}$$

found by the same principle as in Chapter 3. Again to reach to a problem of system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, we have to discretize the problem. The

spatial grid will be divided into N_w equidistant grid points and likewise the angular grid will be divided into N_θ equidistant grid points. Then $\boldsymbol{F}$, the discrete intensity function, will be of dimension $N_w \times N_\theta$. The detections $\boldsymbol{G}_k$ will be of dimension N_y^k dependent on which detector k. This means that each detector can have an individual number of grid points. As in the problem with two spatial dimensions the final step is then to stack both $\boldsymbol{F}$ and $\boldsymbol{G}$ in column vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ and construct the model matrix $\boldsymbol{A}$ by doing a forward operation with unit vectors.

6.2 Test Problems

As described in Section 5.1.2 the Monte Carlo method is using a Gaussian prior, which simply adds a perturbation based on a Gaussian distribution to the previous solution. In Appendix A some basic tests are done using this prior information. The idea is to experiment with the implementation to obtain basic knowledge from this simple set-up. From the results we saw that finding the solutions close to the exact solution might be hard or even impossible. Therefore using all the information about the actual physical set-up might improve the efficiency of the method. We aim to formulate the prior, so realizations will be fairly close to the exact solution.

At first the prior knowledge is based on an assumption about θ. This assumption is of course consistent with what might be possible in the actual set-up. The assumption is described by defining a fixed number of discrete θ values, where w is still uniform distributed. For each θ there is an intensity along the w-axis. In this first attempt only one θ value is chosen. In practice this means that for all pixels on the source the rays are only emitted in one direction.

When setting up the simulation we have to construct an exact solution corresponding to one θ value. When performing the experiments we choose to focus on three types of variation in the intensity. We start simple with the intensity being constant, then slowly varying it and then in the end a random intensity. The latter is actually the most likely outcome of real data. Three different exact solutions are seen in Figure 6.2, where the three intensity outcomes are represented.

What we wish to implement as our prior is the information about the discrete θ value, which is represented in the realizations. The method does not know what this one value of θ is, therefore we give as prior information what we think it is. We choose to specify an interval of three pixels, where the value of θ is possible to be located. So the θ_i of the realization is produced with one of the values

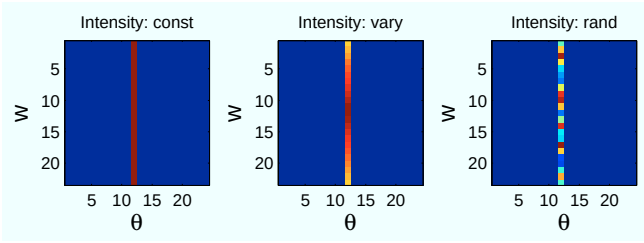

Figure 6.2: Exact solution corresponding to three types of intensity and one discrete θ value.

$\theta_{i-1}, \theta_i, \theta_{i+1}$ with the same probability $\frac{1}{3}$. This way the value of θ cannot take all possible values and thereby the solutions are restricted. We let the intensity perturbate using the gaussian perturbation described in Section 5.3.1.

Since we started by choosing a simple exact solution we now increase the number of columns e.g. the number of discrete θ values, that are present in the exact solution. We choose to look at four different θ values and as above we choose the prior to restrict each value to three possible values. The exact solutions are seen in Figure 6.3. It is simply the same intensity in the spatial coordinate, but now the rays are sent out in four different angles.

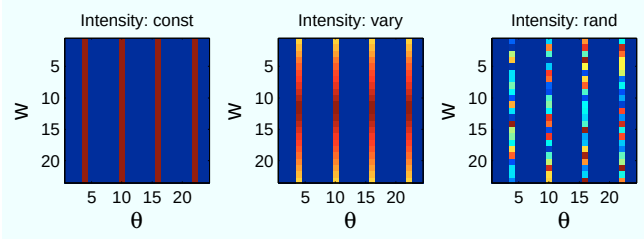

Figure 6.3: Exact solution corresponding to three types of intensity and four discrete θ values.

6.2.1 Strained lattice

In the previous test problems the assumption about the crystal lattice has been, that the grains were perfect orientated within the lattice. To make the simulations more like the real set-up, we now look at a strained lattice. We still assume, that the rays are emitted more likely in a some angles than others, but instead of a ray emitted in just one angle, we now consider that due to the strained

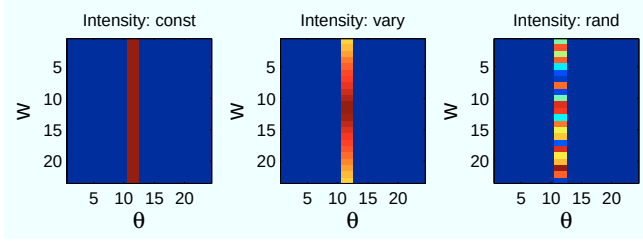

Figure 6.4: Exact solution corresponding to three types of intensity and one discrete θ value.

lattice rays are spread out on a number of neighbor angles. In Figure 6.4 an example of the above described is shown. We notice, that the intensity is the same for all angles in the same neighborhood. This is of course a simplification, but good enough to illustrate the problem.

Since the test problem is now described by one θ value and a varying number identical neighbor values, we have to modify our prior as well. We modify it, so instead of perturbating the θ value in a fixed interval, we now denote two parameters, $\bar{\theta}$ and $\bar{\beta}$, which will be used to find the index of θ and the number of angles the ray is spread out on. The $\bar{\theta}$ value describes the mean value of the value of θ. The second parameter describes the mean value of the number of neigbor angles related to the same intensity. To avoid the values to be negative the two parameters are both described by a lognormal distribution. The procedure is to find the θ value from a lognormal distribution $\ln \mathcal{N}(\bar{\theta}, \sigma^2)$ and the number of angles, the ray is spread out on, also from a lognormal distribution $\ln \mathcal{N}(\bar{\beta}, \sigma^2)$. The standard deviation σ is chosen to be identical to error on data used, when finding the misfit value.

As in the procedure with the test problems with one value of θ in the previous section, we now modify the problem, so instead of working with one discrete θ, we illustrate the problem with four values of θ. This is more realistic and also a more complex problem. The test problems with different intensities are seen in Figure 6.5. We use the same prior as when only one angle was used with the lognormal distributed parameters.

6.3 Reverse Ray Tracing

When doing the forward calculation we exploit the geometry in the experiment. The distances between the detectors and the source along with the maximum

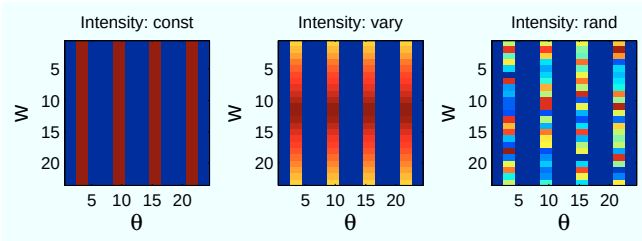

Figure 6.5: Exact solution corresponding to three types of intensity and four discrete θ values.

angle of θ, the sizes of the detectors and sources, summarized in Table 6.1, are used when doing the calculation. Based on the same procedure it is possible to construct a reverse ray tracing algorithm, which given the detections eliminates angle pairs and coordinates at the source, which cannot describe the data. The way to construct the reverse ray tracing algorithm is to start at the far-field detector. Choosing one detection on the detector and another detection at the second near-field detector, a line can be drawn between them. Then the pair of angles can be found based on the straight line through the two detections. Then it is possible to check if this straight line hits the first detector in a detection and if it does, whether it hits the source. So a straight line hitting detections at all detectors and still end up in the source, can then describe a possible ray emitted from the source. Repeating this test through all detections we find a distribution in the source, which then describes which pixels at the source, the cones most likely emit from. This information will be used both as prior information and as starting guess. The reconstructions using this starting guess to the inversion method will be shown in the next section and in Chapter 7 in the section, where the results are discussed.

6.3.1 Revers Ray Tracing in Two Dimensions

Looking at the simple test problem with one value of w and one value of θ, we try to use the reverse ray tracing function to determine which pixels on the source, the light can be emitted from to generate the specific data. This result is illustrated in Figure 6.6, where the exact distribution of w and θ is seen along with the one found by the reverse ray tracing algorithm. It shows that only two values of θ can be obtained to fulfill the detections. In the w-direction it seems that several values of w fulfill the data. The exact data is seen in Figure 6.7 along with the data corresponding to the source distribution found by the reverse ray tracing. The reconstructed data covers the pixels, where the exact data is detected. The intensity is higher, but that is due to the higher number

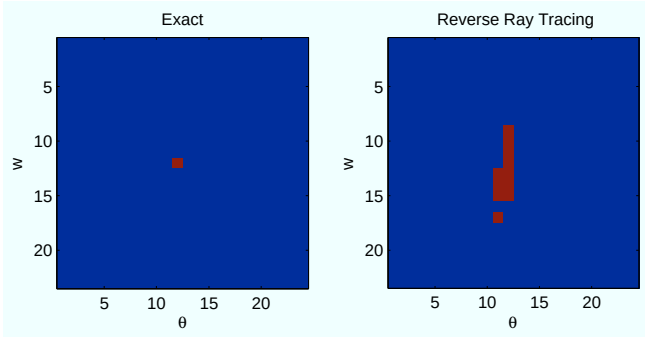

Figure 6.6: Result of the reverse ray tracing algorithm along with the exact test problem.

of active pixels on the source. Based on a source with more light emitted the intensity detected will also be higher. Now we have both a possible starting guess and a distribution of active source pixels, which can be used as prior information.

In this chapter we will mainly use the reverse ray tracing algorithm in relation to a possible starting guess, but in Chapter 7 it will play a more important role.

6.4 Results

In the previous section we described some test problems and their corresponding prior that we want to use with our Monte Carlo simulation. In this section we will analyze the results of the method along with the burn-in period as a function of starting guess. We will mainly focus on using $\boldsymbol{x}_{\text{exact}}$ and a deterministic solution as starting guess. To compare the different solutions a relative error is calculated, based on an average solution described later in this section. This will along with the solution visualized, give a good indication, whether the reconstruction is acceptable or not. Based on different levels of noise and step lengths we will hopefully find some patterns in the errors on the reconstructions.

6.4.1 Single Ray in each Direction

To start simple we look at the case with one value of θ being nonzero and constant intensity along the w axis. We choose to look at different levels of

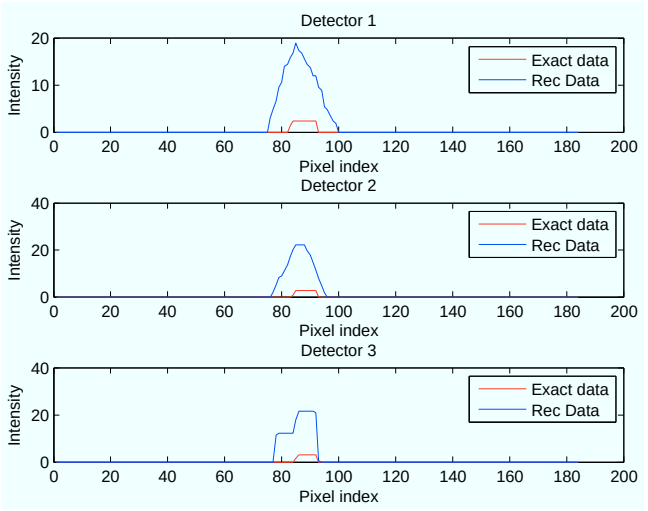

Figure 6.7: The corresponding data detected at the three detectors.

Noise level	0.1	0.01	$1 \cdot 10^{-4}$	$1 \cdot 10^{-8}$
Rel. error	0.1836	0.1495	0.0014	$2.4586 \cdot 10^{-7}$
s	$5 \cdot 10^4$	$3{\cdot}10^5$	$1 \cdot 10^7$	$5{\cdot}10^{13}$

Table 6.2: The relative errors on the solution, the corresponding noise level and step length s using $\boldsymbol{x}_{\text{exact}}$ as starting guess.

Poisson noise. As described in Section 3.3, this is a realistic type of noise in this problem. We want to see what influence the noise level in data has in the reconstructions. When using $\boldsymbol{x}_{\text{exact}}$ as starting guess, the reconstructions seen in Figure 6.8 is obtained. We can conclude along with the values in Table 6.2 that for the data containing a high level of noise $(0.1, 0.01)$ the reconstructions are further away from the exact solution, than when adding small amounts of noise. When using data containing small amounts of noise, we can see from the misfit plot in Figure 6.10, that the acceptance rate in the method is very low. Actually it is around 1 percent. This is verified by the theory, which states that when a small amount of noise is added, the distribution of the solution narrows, making it very hard to find a distribution for the solution using the Monte Carlo method. The method rejects almost all realizations, because the residual becomes large. This verifies that the Monte Carlo method performs best with a significant amount of noise on data. The reason for the low relative error is that the exact solution is used at starting guess, so the realizations are not accepted if they are too far away from the exact solution.

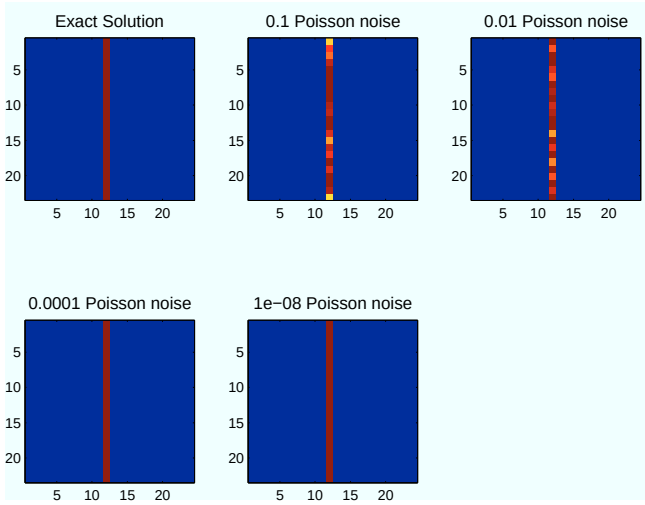

Figure 6.8: Exact solution along with four reconstructions with different levels of Poisson noise on data, and using $\boldsymbol{x}_{\text{exact}}$ as starting guess.

Adding noise to data means that we add noise on each detector. This is illustrated in Figure 6.9 with 10 % Poisson noise added to data.

If we look at the misfit plots in Figure 6.10, we see that the level of the misfit values is close to 10. If the level was smaller it would be problematic, since the solutions should not be the exact solutions but a distance corresponding to σ away. We can also conclude that the burn-in period is very short with the first three levels of noise. This is due to the use of $\boldsymbol{x}_{\text{exact}}$ as starting guess.

We have now analyzed one test problem and seen that the method solved the problem as expected. It would be interesting to see how the method behaves, when another starting guess is used. We continue our experiments with the same test problem, but just using a deterministic solution as starting guess instead.

We have chosen to look at the three deterministic methods Kazmarcz (ART), Cimmino and Conjugate Gradient Least Squares (CGLS). In Figure 6.11 the solutions from the three methods after 20 iterations are shown. We see that especially the CGLS method seems to make a good reconstruction. The deterministic methods perform well due to the constant intensity. The last figure shows a weighted mean, which is used as a starting guess to the stochastic method.

Using this starting guess we get the reconstructions visualized in Figure 6.12

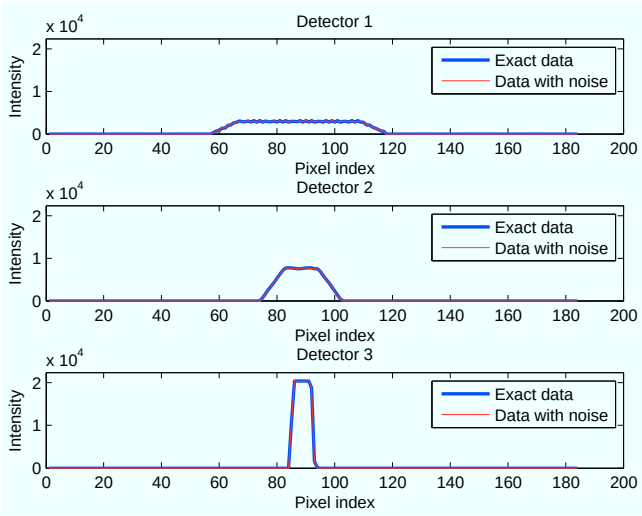

Figure 6.9: Data on the three detectors along with the noisy version, when 10 % noise is added.

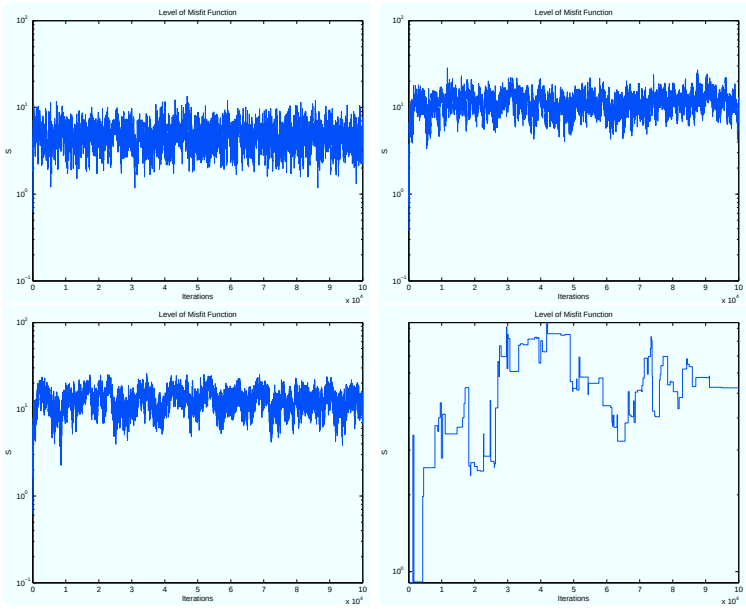

Figure 6.10: Misfit functions for different levels of noise on data and using the exact solution as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

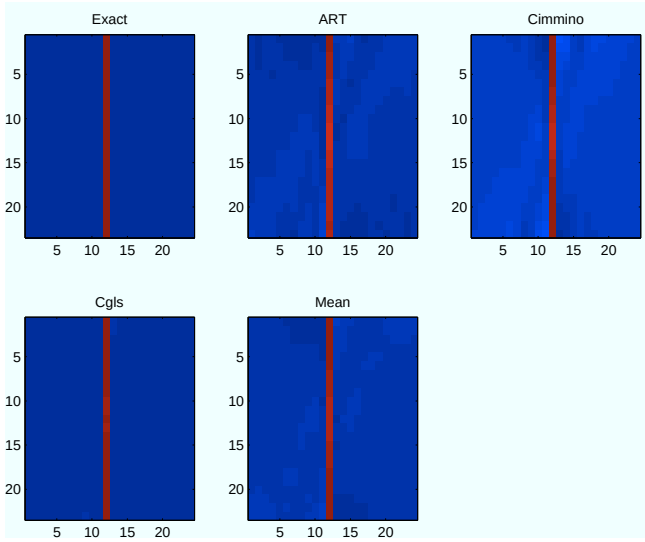

Figure 6.11: Three deterministic solutions and a weighted mean of those along with the exact solution.

Noise level	0.1	0.01	$1 \cdot 10^{-4}$	$1 \cdot 10^{-8}$
Rel. error	0.3716	0.2973	0.0091	0.0087
s	$5 \cdot 10^4$	$3{\cdot}10^5$	$1 \cdot 10^8$	$1{\cdot}10^{16}$

Table 6.3: The relative errors on the solution, the corresponding noise level and step length s using a deterministic solution as starting guess.

and the relative errors in Table 6.3. We see that the relative errors increase marginally, and from the burn-in periods in Figure 6.13 we can conclude that with this test problem the difference in the results using different starting guesses is small. This is due to the deterministic solution, which is very close to the exact solution. We need to investigate the results, when using more complex problems to make any conclusions. We also see that the level, where the misfit values level out is approximately the same as using the exact solution as starting guess.

Now we want to use the solution from the reverse ray tracing algorithm described in 6.3 as starting guess. As above we use four different levels of Poisson noise and the results are seen in Figure 6.14 and 6.15 and Table 6.4. We see that the reconstructions are acceptable also with this starting guess.

What would be interesting to look at, is whether the burn-in period is dependent

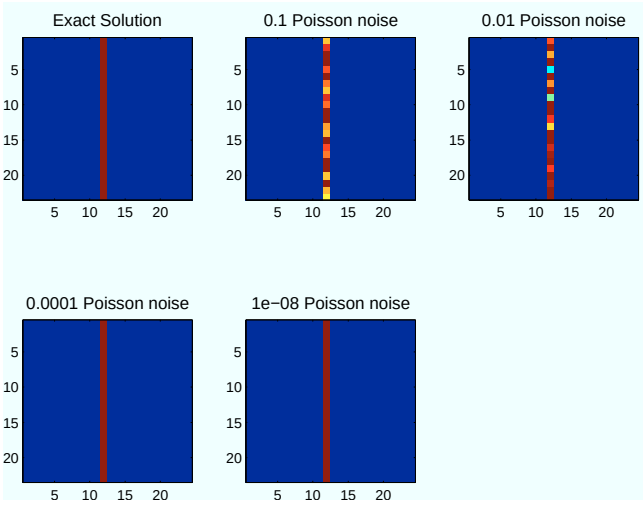

Figure 6.12: Exact solution along with four reconstructions with different levels of Poisson noise on data, and using a deterministic solution as starting guess.

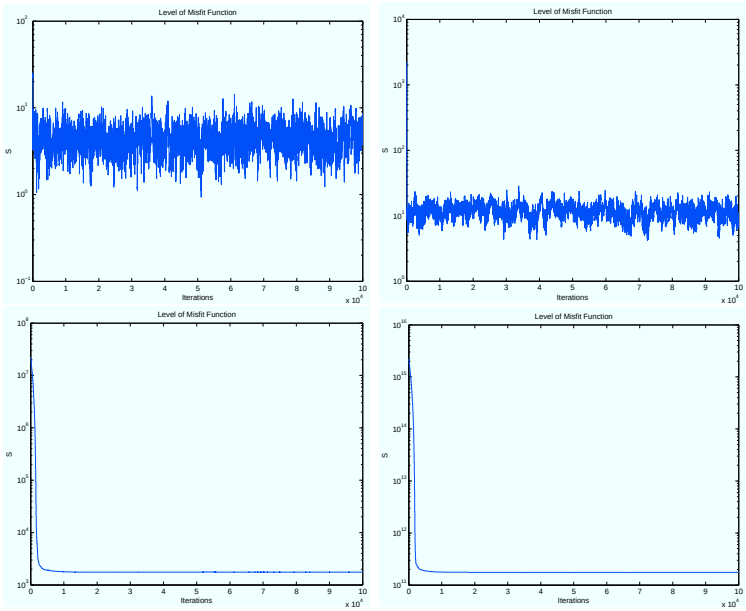

Figure 6.13: Misfit functions for different levels of noise on data and using a deterministic solution as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

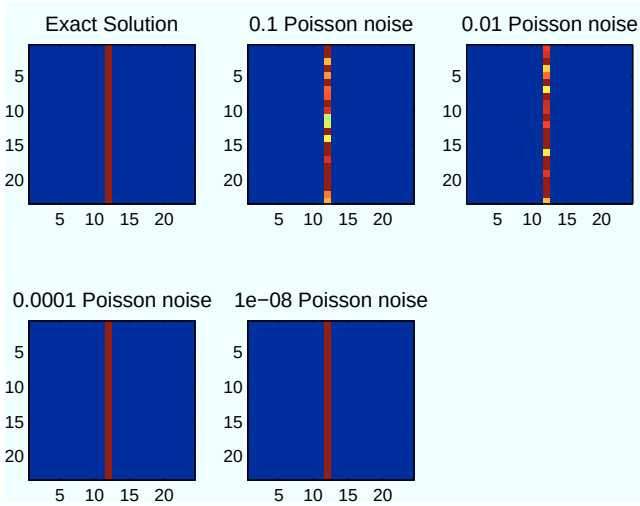

Figure 6.14: Exact solution along with four reconstructions with different levels of Poisson noise on data, and using a deterministic solution as starting guess.

Noise level	0.1	0.01	$1 \cdot 10^{-4}$	$1 \cdot 10^{-8}$
Rel. error	0.6467	0.2410	0.0026	$3.5535 \cdot 10^{-7}$
s	$7 \cdot 10^2$	$7 \cdot 10^3$	$1 \cdot 10^7$	$1 \cdot 10^{13}$

Table 6.4: The relative errors on the solution, the corresponding noise level and step length s using a deterministic solution as starting guess.

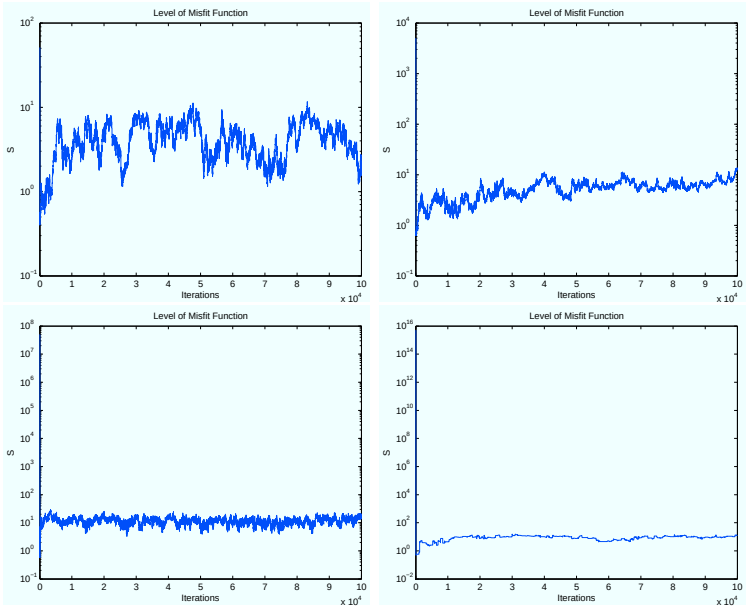

Figure 6.15: Misfit functions for different levels of noise on data and using a reverse ray tracing as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

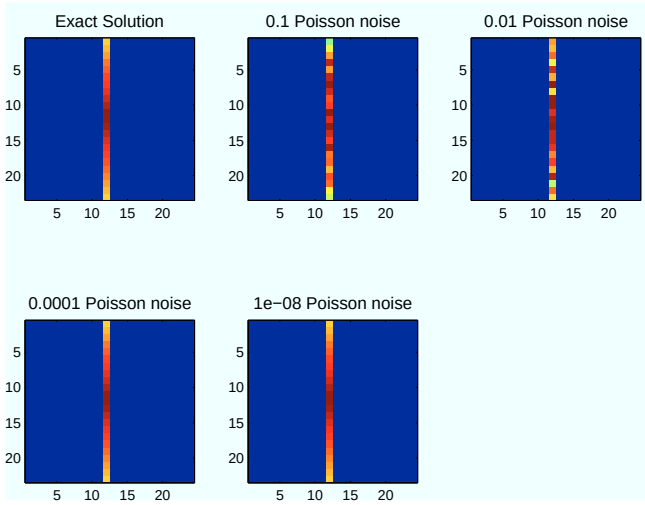

Figure 6.16: Exact solution along with four reconstructions with different noise levels on data, and using $\boldsymbol{x}_{\text{exact}}$ as starting guess.

Noise level	0.1	0.01	$1 \cdot 10^{-4}$	$1 \cdot 10^{-8}$
Rel. error	0.1393	0.1397	0.0014	$4.5585 \cdot 10^{-7}$
s	$5 \cdot 10^{4}$	$3 \cdot 10^{5}$	$1 \cdot 10^{7}$	$5 \cdot 10^{13}$

Table 6.5: The relative errors on the solution, the corresponding noise level and step length s using $\boldsymbol{x}_{\text{exact}}$ as starting guess.

on the complexity of the test problem. We can modify our test problem with both the intensity and the number of discrete θ. First we focus on the intensity. Looking at the reconstructions of the problem with slowly varying intensity in Figure 6.16, we see that the reconstructions are further away from the exact solution. This is verified by the relative errors in Table 6.5. Only the error, where the noise level is 10^{-8}, seems to decrease. Though looking at the corresponding misfit plot in Figure 6.17, we see that acceptance rate is very low compared to the other misfit plots in the figure. This is due to the small amount of noise in data. Since σ is very small, the misfit values will often be extremely large, and therefore the realizations will often not be accepted. But if a realization is accepted, it is most likely because it is very close to the exact solution. Therefore we have a small relative error. From the misfit plots in the figure, we can see that the burn-in periods are relatively short.

Again we want to compare these results with the results using a different starting guess. This time the deterministic solutions are not as close to the exact, so we

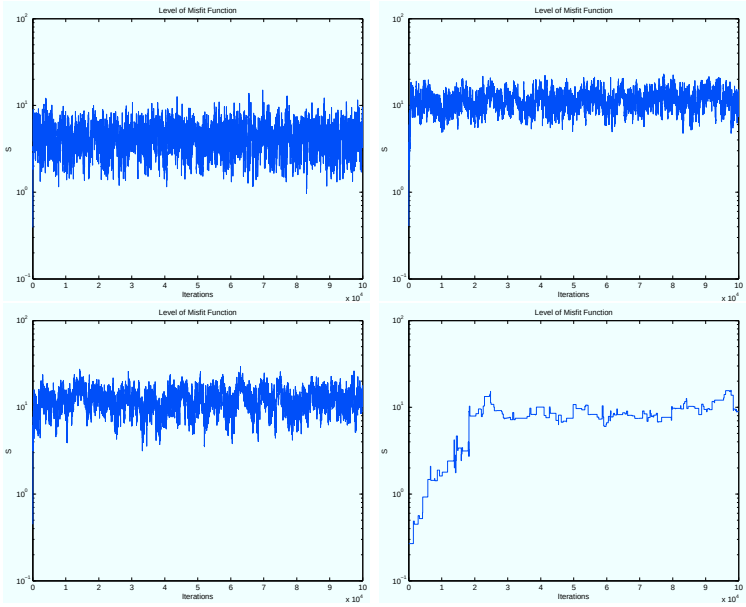

Figure 6.17: Misfit functions for different levels of noise on data and using the exact solution as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

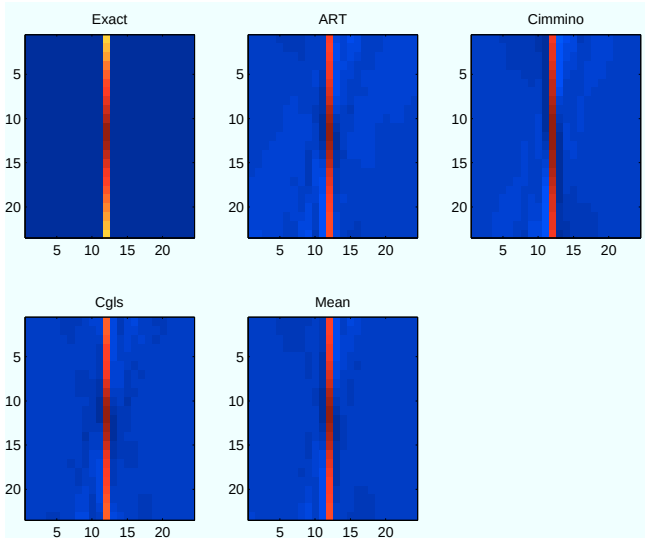

Figure 6.18: Deterministic solutions and the starting guess used.

Noise level	0.1	0.01	$1 \cdot 10^{-4}$	$1 \cdot 10^{-8}$
Rel. error	0.2402	0.2198	0.0224	0.0217
s	$5 \cdot 10^4$	$3{\cdot}10^5$	$5 \cdot 10^6$	$5{\cdot}10^{14}$

Table 6.6: The relative errors on the solution, the corresponding noise level and step length s using a deterministic solution as starting guess.

might expect different results - see Figure 6.18. The reconstructions, the errors and the misfit plots are seen in Figure 6.19, Table 6.6 and Figure 6.20. As expected the burn-in periods increased for the two cases with lowest amount of noise present. It makes sense, that the burn-in period increases with complexity.

Continuing with analyzing the results, the next step is to increase the number of discrete angels, θ, to four. We look at the problem with random intensity. We have now seen that, how the method deals with different levels of noise. When a small amount of noise is added, the relative error decrease. In these simulations the solution methods are able to handle low amounts, but it is also possibility that the realizations are not accepted at all. We will see examples of that in the next chapter. Therefore we now look at the same noise level, 10 %, which we know is a realistic level, with two different type of noise, Gaussian and Poisson noise The reconstructions are seen in Figure 6.21 and 6.22 using the exact solution and a deterministic solution respectively. The corresponding

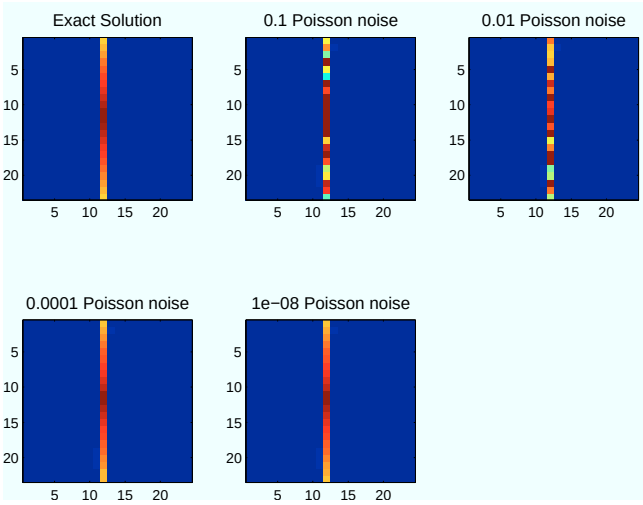

Figure 6.19: Exact solution along with four reconstructions with different noise levels on data, and using a deterministic solution as starting guess.

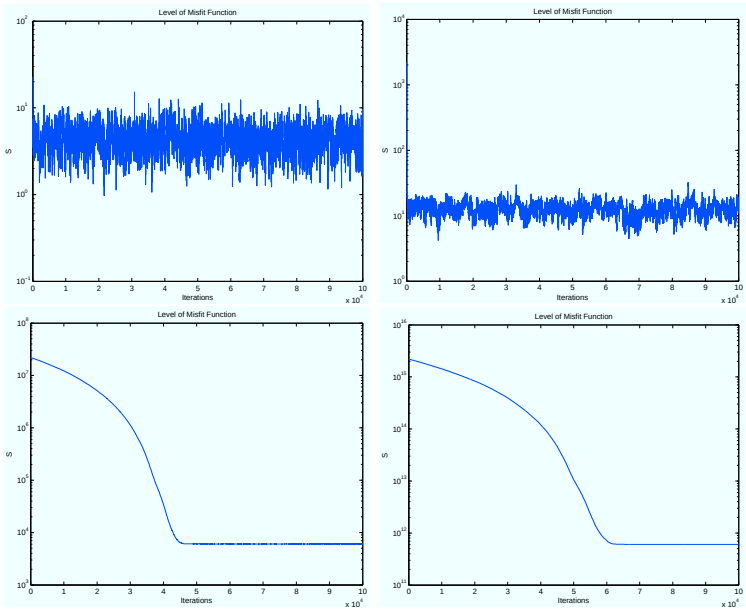

Figure 6.20: Misfit functions for different levels of noise on data and using a deterministic solution as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

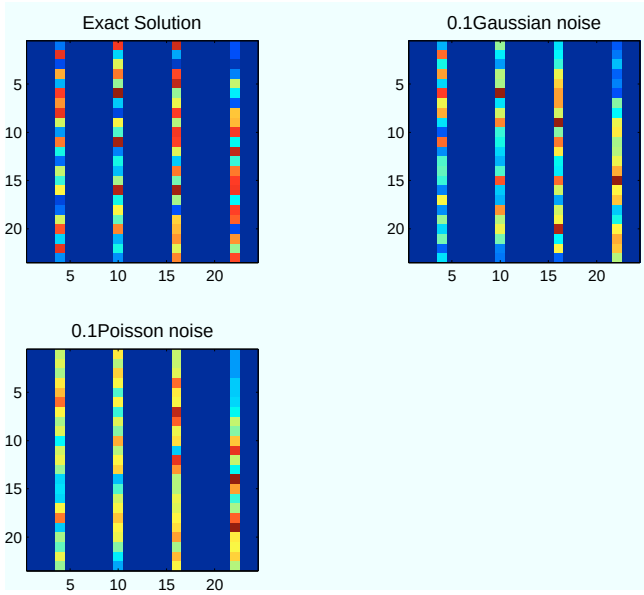

Figure 6.21: Exact solution along with four reconstructions with different noise levels on data, and using the exact solution as starting guess.

Noise type	Gaussian	Poisson
Noise level	0.1	0.01
Rel. error	0.4964	0.4581
s	$5 \cdot 10^4$	$1{\cdot}10^5$

Table 6.7: The relative errors on the solution, the corresponding noise level and type, and step length s using $\boldsymbol{x}_{\text{exact}}$ as starting guess.

errors are found in Table 6.7 and 6.8 and with the deterministic solution used as starting guess in Figure 6.24. One important thing to notice about the deterministic solutions is how bad the reconstructions are compared to the case with constant intensity. This also affects the relative errors. The misfit plots are found in Figure 6.23 and 6.25 using the exact solution and a deterministic solution respectively. As the number of θ increases, it is obvious that the relative error increases and also that the relative errors using the deterministic solution as starting guess are higher than when using $\boldsymbol{x}_{\text{exact}}$. We see that the burn-in period is still relatively short. The burn-in period is very dependent on the choice of step length s. If s is small, the burn-in period becomes large, but if s is too big, the method might find solutions too far away from the exact solution.

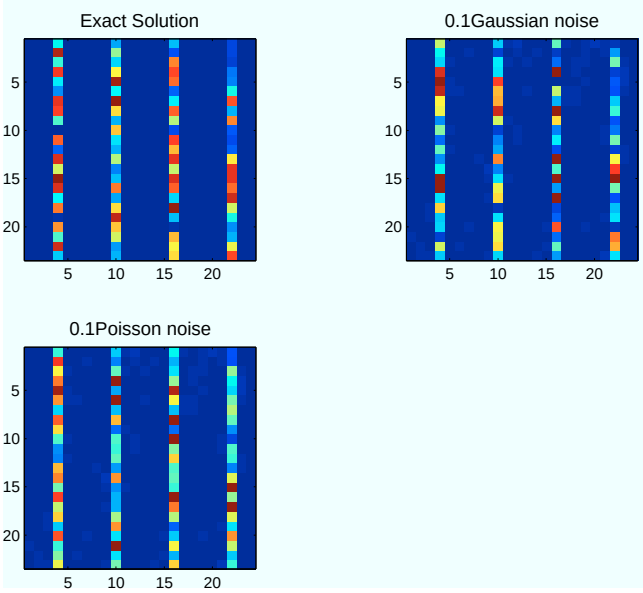

Figure 6.22: Exact solution along with four reconstructions with different noise levels on data, and using a deterministic solution as starting guess.

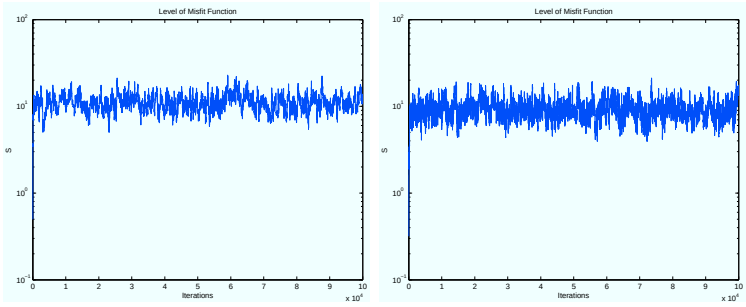

Figure 6.23: Misfit functions for different types of noise. Left Gaussian noise and right Poisson noise using the exact solution as starting guess.

Noise type	Gaussian	Poisson
Noise level	0.1	0.1
Rel. error	0.7408	0.5905
s	$5 \cdot 10^4$	$1{\cdot}10^5$

Table 6.8: The relative errors on the solution, the corresponding noise level and type, and step length s using a deterministic solution as starting guess.

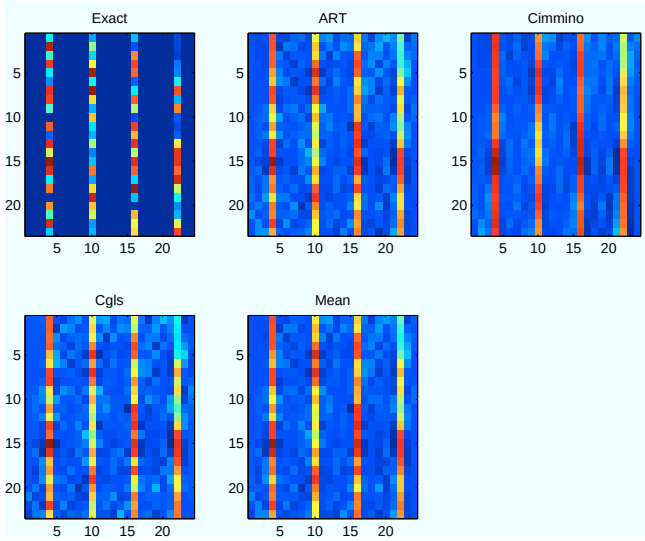

Figure 6.24: Deterministic solutions and the starting guess.

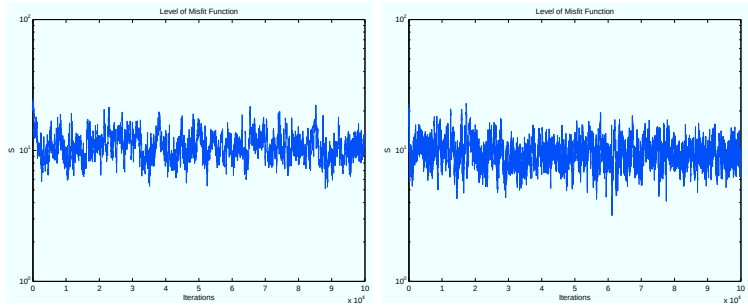

Figure 6.25: Misfit functions for different types of noise. Left Gaussian noise and right Poisson noise using a deterministic solution as starting guess.

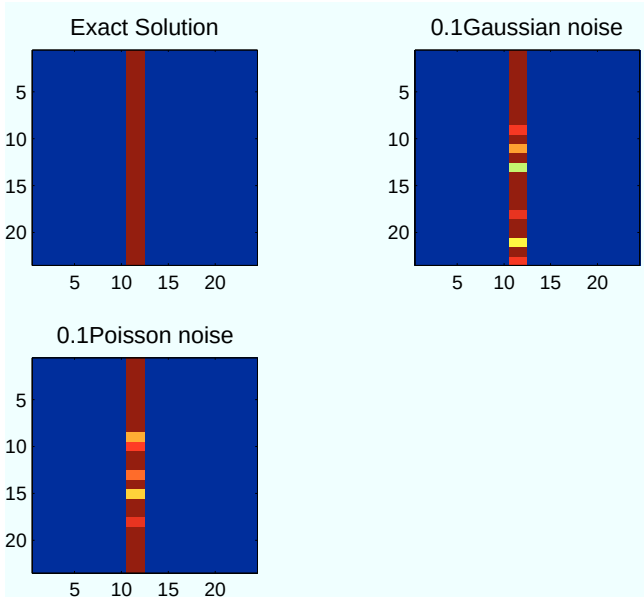

Figure 6.26: Exact solution along with two reconstructions with different noise on data, and using the exact solution as starting guess.

Noise level	0.1	0.1
Rel. error	0.2116	0.2143
s	$5 \cdot 10^3$	$1{\cdot}10^4$

Table 6.9: The relative errors on the solution, the corresponding noise level and type, and step length s using the exact solution as starting guess.

6.4.2 Multiple Rays in each Direction

As described in Section 6.2.1, when the crystal lattice is compressed each ray will be spread out on a number of neighbor angles. Using a test problem where the rays are spread out on two values of θ will now be analyzed. As in the section above we start simple and then increase the complexity. Using constant intensity for one ray and the exact solution as starting guess gives the reconstructions seen in Figure 6.26. Again the stochastic method seems to be able to reconstruct the exact solution. Looking at the errors in Table 6.9, we can conclude that the relative errors increase compared to the problem where the ray is represented by only one value of θ. The misfit plots in Figure 6.27 with 10 % noise has a short burn-in period, as we saw earlier.

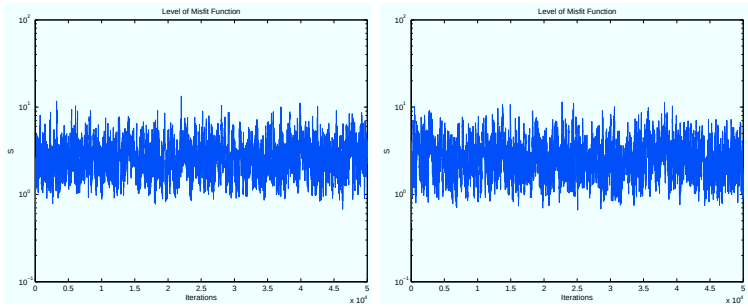

Figure 6.27: Misfit functions for different types of noise. Left Gaussian noise and right Poisson noise using the exact solution as starting guess.

Noise type	Gaussian	Poisson
Noise level	0.1	0.1
Rel. error	0.2225	0.2384
s	$5 \cdot 10^3$	$1{\cdot}10^4$

Table 6.10: The relative errors on the solution, the corresponding noise level, noise on the method and step length s using a deterministic solution as starting guess.

Now we want to compare the results by using a deterministic solution as starting guess. The reconstruction is seen in Figure 6.28. Along with the errors in Table 6.10 and the misfit plots in Figure 6.29 we see that the errors increase, but the burn-in period still remains short.

Increasing the number of columns as in the previous section is the next step. By doing that we expect a reconstruction further away from the exact solution, as the problem gets more complex. Looking at Figure 6.30 and Table 6.11 we see that the quality of the reconstruction using a deterministic solution as starting guess decreases. Also using $\boldsymbol{x}_{\text{exact}}$ as starting guess results in increase relative errors and reconstructions further away from the exact solution - see Figure 6.32 and Table 6.12. The burn-in periods in Figure 6.31 and 6.33 are still very short with this high level of noise. The rest of the simulations are not included, since the results did not vary much from the ones already shown. The results of the reconstruction were similar with the ones in the previous section just marginally worse, and the relative errors were slightly higher.

Summing up on the results we noticed some important aspects of the method. First of all it seems that the method is sensitive towards the choice of starting guess regarding the length of the burn-in period. General for using a Monte Carlo simulation is that it can be hard to find the solutions lying close to the

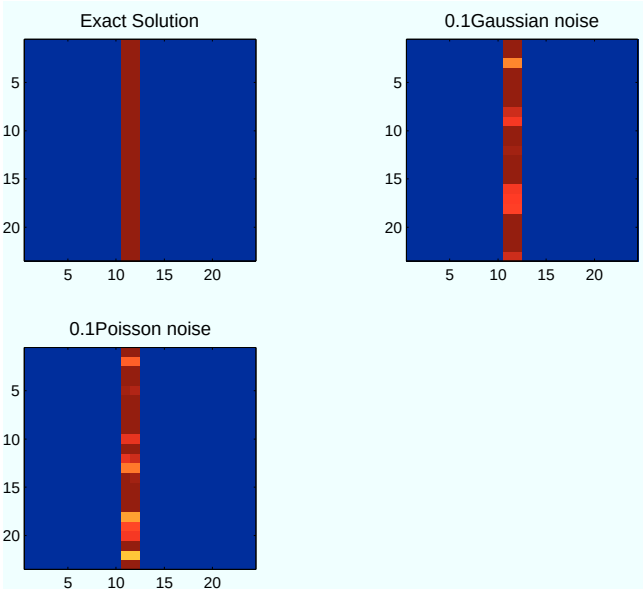

Figure 6.28: Exact solution along with two reconstructions with different noise on data, and using a deterministic starting guess.

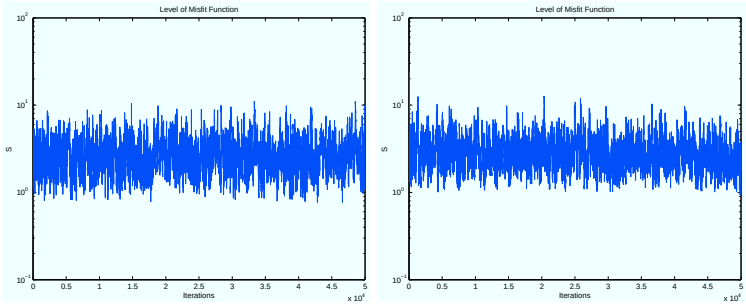

Figure 6.29: Misfit functions for different types of noise. Left Gaussian noise and right Poisson noise using a deterministic solution as starting guess.

Noise type	Gaussian	Poisson
Noise level	0.1	0.1
Rel. error	0.5623	0.4710
s	$1 \cdot 10^4$	$1{\cdot}10^4$

Table 6.11: The relative errors on the solution, the corresponding noise level, noise on the method and step length s using a deterministic solution as starting guess.

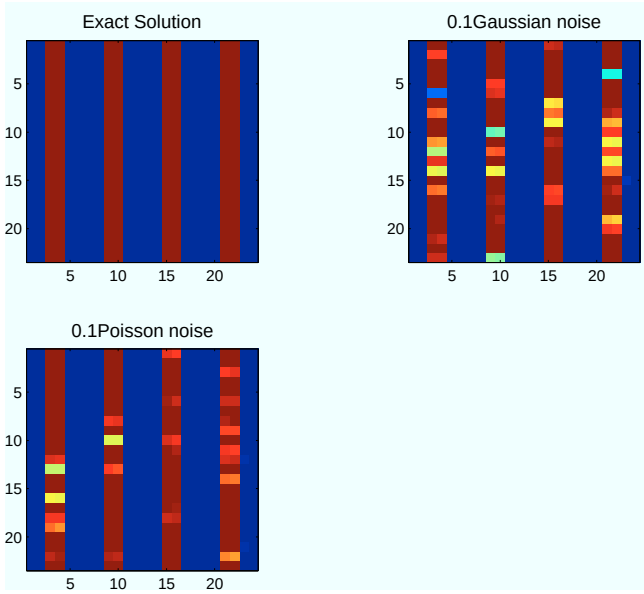

Figure 6.30: Exact solution along with five reconstructions with different noise levels on data, and using a determistic solution as starting guess.

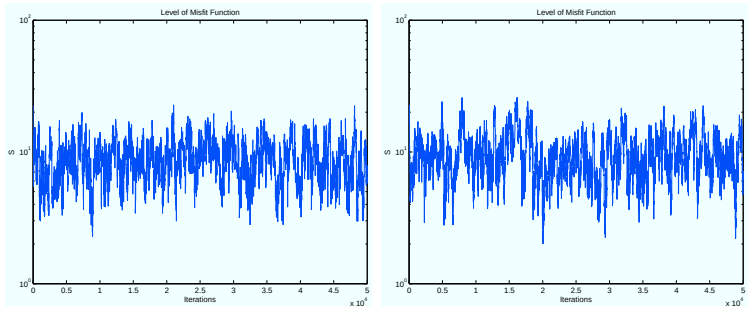

Figure 6.31: Misfit functions for different levels of noise on data and using a deterministic solution as starting guess. Top left: 10 % Poisson noise, top right: 1 % Poisson noise, bottom left: $1 \cdot 10^{-2}$ % Poisson noise and bottom right: $1 \cdot 10^{-6}$ % Poisson noise.

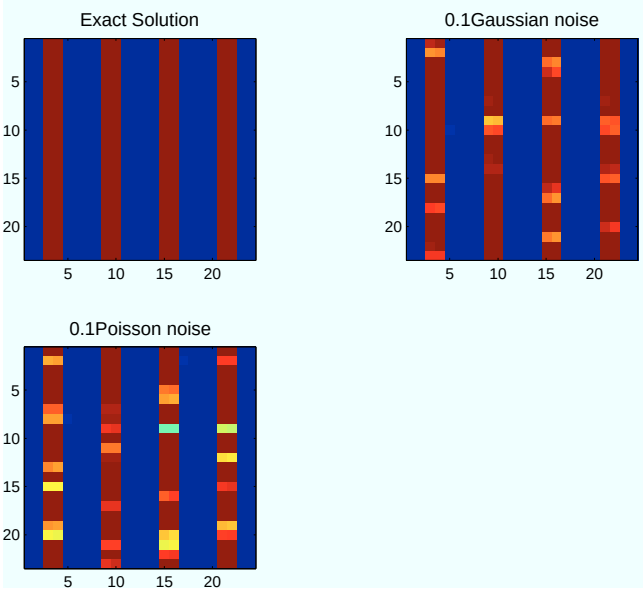

Figure 6.32: Exact solution along with two reconstructions with different type of noise on data, and using the exact solution as starting guess.

Noise type	Gaussian	Poisson
Noise level	0.1	0.1
Rel. error	0.3933	0.4878
s	$5 \cdot 10^3$	$1{\cdot}10^4$

Table 6.12: The relative errors on the solution, the corresponding noise level, type of noise and step length s using the exact solution as starting guess.

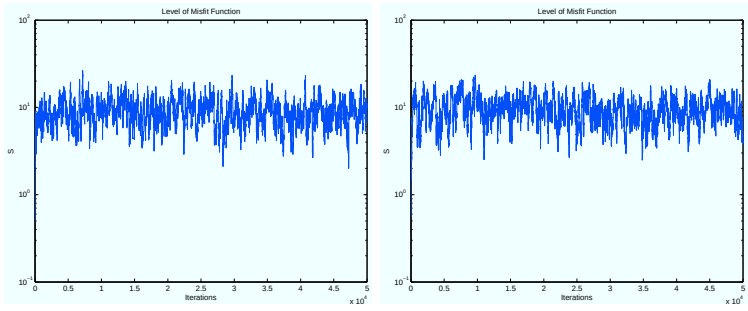

Figure 6.33: Misfit functions for different types of noise. Left Gaussian noise and right Poisson noise using a deterministic solution as starting guess.

exact solution, what we call the solutions after burn-in period. Of course the starting guess has a great influence on how easily we find these solutions. Therefore it is very important to choose a starting guess with care. The amount of noise we add on data also plays a huge role in the reconstructions. As we saw in the first reconstruction, when the ray is only spread out on one θ value the burn-in period increased as the noise level decreased. This is due to the characteristic of the Monte Carlo method. When a small amount of noise is added the density distribution of the solution can be very hard to find.

Also it was clear that increasing the complexity, decreased the quality of the solution. This is obvious of course, but still problematic, since the problems we solve are fairly simple and using the real world data imply a much more complex problem. One last thing to mention is the step length s. It contributes to length of the burn-in period. If we choose a small step length the burn-in period will be large, because it takes more iterations to reach realizations, which describes the solution. The step length also indicates how many of the realizations, which are accepted. The smaller the step length the more realizations accepted. Using Monte Carlo simulation we always aim for a reasonable acceptance rate, often between 30-60 % - see [2].

6.5 Forward Calculation

As mentioned in Section 5.2.1, one of the advantages of using a stochastic method is the possibility of avoiding the multiplication with the matrix $\boldsymbol{A}$. Each time a misfit value is calculated, the residual $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$ is also calculated. So a multiplication with model matrix $\boldsymbol{A}$ happens twice in every Monte Carlo iteration. The size of $\boldsymbol{A}$ as described in Section 2, is based on the size of the detectors and the discrete grid we divide the source plane into, so the size of $\boldsymbol{A}$ is $N_\theta \times N_w \times 3N_d$. This means that $N = N_\theta \times N_w = 3N_d$, when we want to look at a square matrix of dimensions $N \times N$. When the grid is refined, the construction of the model matrix will be very time consuming. In Figure 6.34 the computation times are shown for both the matrix calculation and the forward calculation without matrix. It is possible to calculate the detection at each pixel at each detector, when the intensity distribution is known. The realization containing some discrete values of θ different from zero will then be used to find the corresponding detections on the three detectors. In this way the residual can be found without use of the matrix $\boldsymbol{A}$.

A linear regression in a loglog scale is performed. It is important to notice, that the computation times decrease, when calculating $\boldsymbol{A}$ continually, for the first couple of N's. This is due the behavior of built-in MATLAB functions. Similarly

the same calculation is done for the capacity, each method requires in Figure 6.35. We see that when N grows, the computation time of constructing $\boldsymbol{A}$ also increases with $O(N^2)$. This is verified by the MATLAB code in Appendix C, where the matrix $\boldsymbol{A}$ is constructed. Running through all unit vectors and detector pixel we obtain $N_\theta \times N_w \times 3N_d$. Inside this loop $30N_w$ flops are performed, and it can be approximated as $constant \cdot N^2$ flops.

When the matrix $\boldsymbol{A}$ is computed the multiplication $\boldsymbol{Ax}$ is not very time consuming. This is illustrated in Figure 6.36. We see that constructing the matrix is very time consuming. Therefore it is important to consider if you need to construct various $\boldsymbol{A}$ matrices or just a few. If you only need one $\boldsymbol{A}$ it might be a good idea to construct it and then use the simple modification. But using the matrix is only a possibility, when the problem is not large-scale.

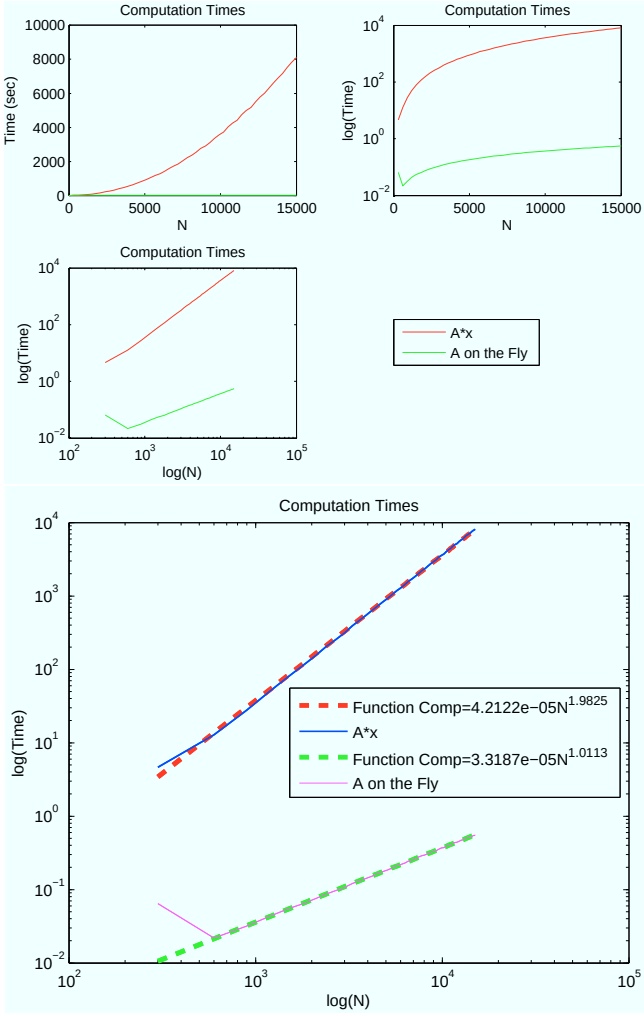

Figure 6.34: Computation times. Top figure shows the data in a linear, logarithmic and double logarithmic plot. The bottom figure shows the regression of the data.

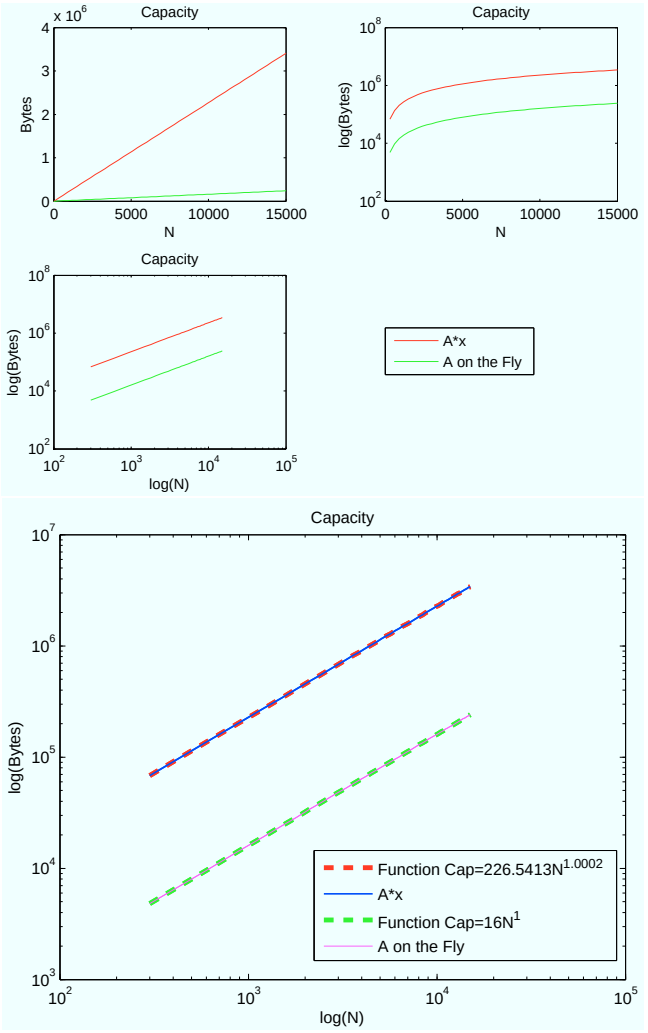

Figure 6.35: Capacity. Top figure shows the data in a linear, logarithmic and double logarithmic plot. The bottom figure shows the regression of the data.

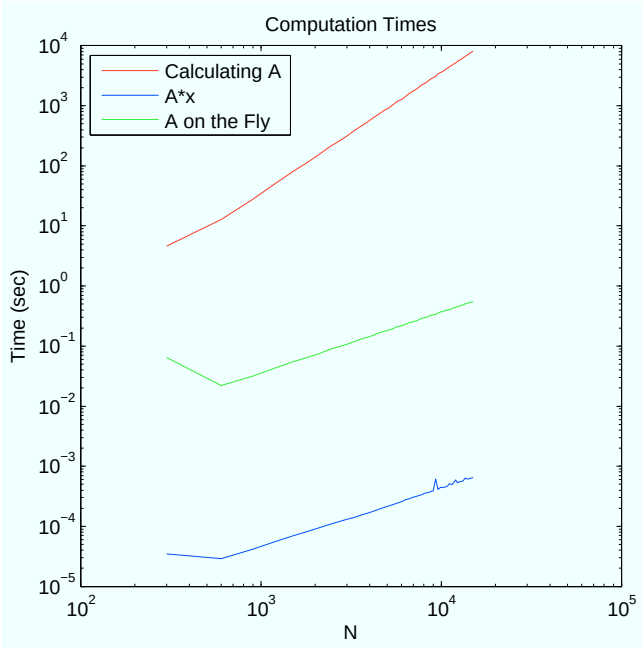

Figure 6.36: Computation times in double logarithmic plot for both the the function, which creates the matrix, the multiplication $\boldsymbol{Ax}$ and the forward calculation without $\boldsymbol{A}$.

Chapter 7

Four-dimensional Problem

In the previous chapter we made a thorough analysis of a simplified problem of the mathematical model described in Section 3.2. We acquired knowledge about the importance of starting guesses, prior information, step length and noise level on data. We now want to conduct a similar analysis on the four-dimensional problem. Again we need to define some test problems based on simulated data, so we can evaluate the performance of the method. As in previous chapter, the procedure will be to investigate the problems of increasing complexity. In the end a complex problem is considered, in order to explore the robustness of the method and also see if the method can be used for more than just simple test problems. The visualization will be different, since the solution will be a four-dimensional matrix. Still we will use a mean of all solutions as a measure of the quality of the method.

7.1 Test Problems

As in Chapter 6 with one spatial dimension we dealt with a couple of well-defined test problems. In this larger problem we can use some of the aspects we explored in the simplified problem, the burn-in period, the noise sensitivity etc. Since each exact solution is represented by the four-dimensional array $\boldsymbol{F}$, it is not as simple to visualize as in the simpler case. If we imagine a simple

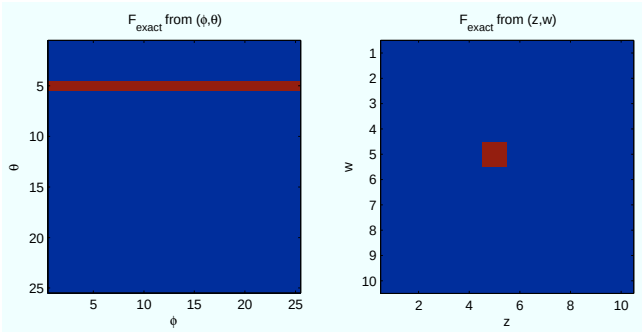

Figure 7.1: Test problem with one active pixel on source where one cone of light is emitted with no variation in the ϕ. Cross sectional plots corresponding to the spatial coordinates (5,5) in the left figure and the pair of angles (5,10) at the right figure.

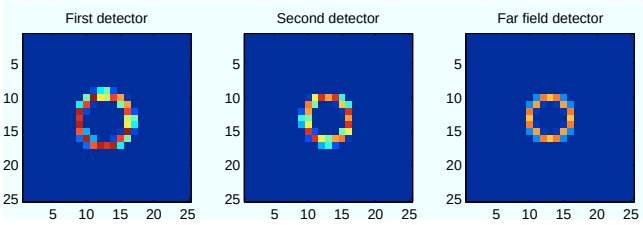

Figure 7.2: The corresponding data detected at the three detectors.

case, where one cone of light is emitted from one point on the source plane, it can be visualized in different ways. One way is to make cross sectional figures of different layers of this cone, this is illustrated in Figure 7.1. In the image to the left we see a cross section in the (ϕ, θ) direction and it is clear that only one value of θ is nonzero and that there is no variation in ϕ. This is verified by the constant intensity along ϕ. The image to the right represents a cross section of the layer at the source, so there we see that light is only emitted from one pixel on the source.

The corresponding data, which is detected at the three detectors is seen in Figure 7.2.

This simple test problem gives rise to many other test problems. We can increase the number of discrete θ values, the number of active pixels on the source and change the variation in ϕ. Taking it from a realistic point of view, ϕ will most likely not have a constant intensity along the cone. It will often consist of different blobs of intensity along the circle on each detector. When ϕ is not

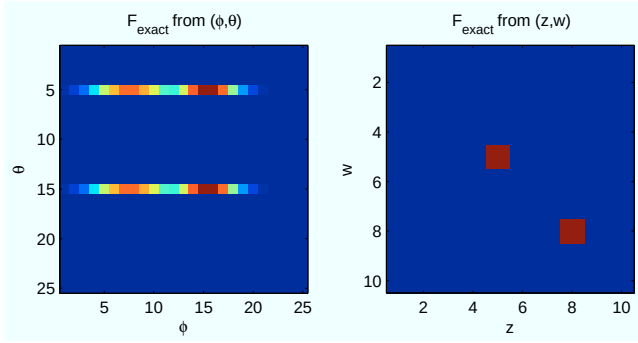

Figure 7.3: The corresponding data detected at the three detectors.

constant it might look like what is illustrated in Figure 7.3. In this example the variation in ϕ is identical for the two cones of light. This is not always the case.

To investigate the robustness of the method it is not enough to look at different test problems. It is also necessary to adjust the modules related to the method. From the previous chapter we discovered how much impact the starting guesses have on the outcome of the stochastic method. Therefore it is essential to use several starting guesses. The exact solution will be used to verify the that the method actually works, and then afterwards we will not use the exact solution as starting guess, since it will not be available, when dealing with real experimental data. We want to show that the method can solve the inverse problem using a starting guess consisting of zeros. But also using a deterministic solution, a solution obtained with the reverse ray tracing calculation and a combination of the last two as starting guess will be tested.

Another important aspect which arises from the knowledge of the scientists conducting the experiments is the variation in ϕ. Since the most realistic case is that ϕ is varying, it is important that our method can handle this. Besides the starting guess we want to investigate how the methods perform with different choices of prior information. We can choose a wide prior distribution, where the distribution of active source is unknown, but we can also utilize the knowledge, we obtain using the reverse ray tracing algorithm about which pixels on the source the rays are most likely emitted from. From the function we obtain a possible distribution of active pixels on the source, which we can convert into prior information.

We then have a supported guess on which pixels, we think, the rays are most likely emitted from. In Figure 7.4 the distribution achieved by using the reverse

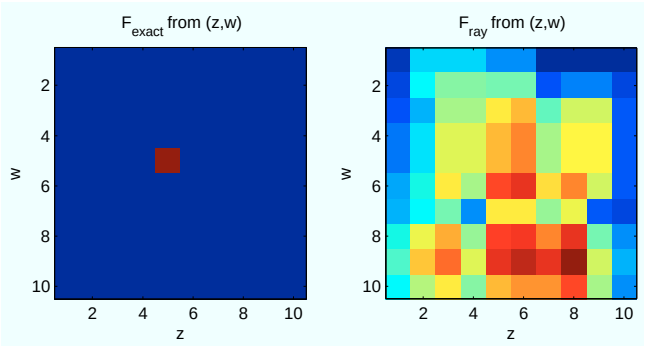

Figure 7.4: Distribution of pixels at the source and the output from the reverse ray tracing algorithm.

ray tracing is seen. This will be used as prior, when there is no variation along ϕ. When we work with variation along ϕ, it might not be enough to specify the possible distribution of pixels at the source found by reverse ray tracing algorithm. Later in this chapter we investigate the performance of the method if the active pixels on the source are known and used as prior information. We build our prior as hard constraints on the problem. As the simple prior in Chapter 6 we construct the prior so it perturbates on the intensity. In this chapter we deal with four parameters that variate namely w, z, ϕ, θ. So it randomly chooses between the parameters and then updates on the current intensity.

Using the reverse ray tracing algorithm on the test problems, we obtain a starting guess, where several angle pairs are possible, but since we know the distribution of discrete θ values, some can be neglected. Therefore we implement an extra prior information, when we use the starting guess obtained from the reverse ray tracing algorithm. This prior indicates that all other θ angles than the discrete ones specified as prior knowledge should be set to zero.

7.2 Results

As in the previous chapter we discuss the results, when we show them. Again we start simple and then advance in complexity. Starting with the simple test problem shown in the previous section we use different starting guesses, which it comprehensive. We start by using x_{exact} as starting guess to verify if the method works. In Figure 7.5 a cross sectional plot is seen for the simple test problem with 10 % Gaussian noise and using the prior information from the reverse ray tracing to find a distribution of the active pixels on the source. The

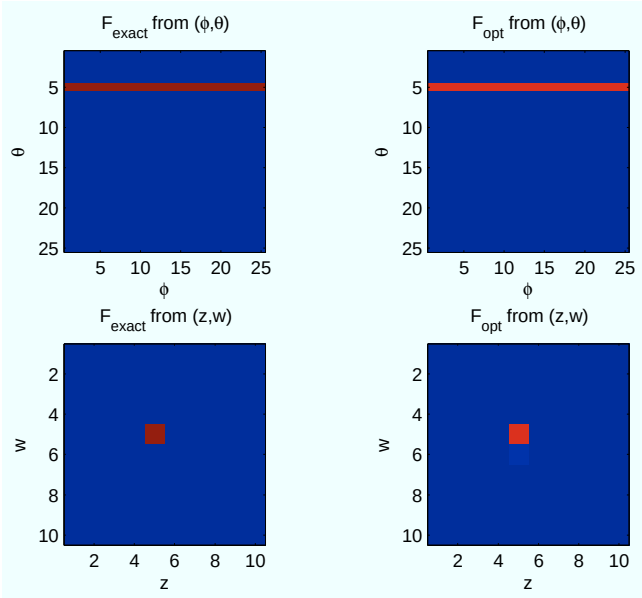

Figure 7.5: Cross sectional plot from the point (5,5) and the angle pair (5,5) using x_{exact} and 10 % Gaussian noise.

relative errors for this test case, with no variation in ϕ, are summarized in Table 7.1.

The method seems to be able to solve the problem using the exact solution as starting guess. We will now show its performance using other starting guesses, starting with a deterministic solution. We have used the solution after 20 iterations with an ART method as the starting guess. The reconstruction, which we will use as starting guess is seen in Figure 7.6. We see that the method is able to reconstruct the discrete θ, but the distribution at the source is more blurred. The reconstruction are seen in Figure 7.7 and Figure 7.8 with 10 % and 1 %

Noise level	0.1	0.01
Rel. error on solution	0.0896	0.0072
Rel. error on data	0.0866	0.0049
Step length	$5 \cdot 10^6$	$5{\cdot}10^6$

Table 7.1: The relative errors on the solution and the corresponding data based on a mean of all solutions, the corresponding step length s using the $\boldsymbol{x}_{\text{exact}}$ as starting guess.

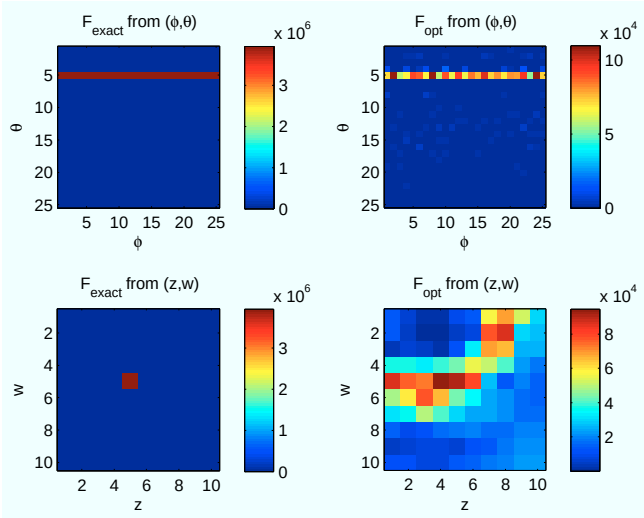

Figure 7.6: Cross sectional plot from the point (5,5) and the angle pair (5,5) showing the deterministic solution for both 10 % Gaussian noise.

Noise level	0.1	0.01
Rel. error on solution	0.5585	0.5790
Rel. error on data	0.4020	0.3657
Step length	$5 \cdot 10^7$	$1{\cdot}10^6$

Table 7.2: The relative errors on the solution and the corresponding data based on a mean of all solutions, the corresponding step length s using a solution found by ART method as starting guess.

Gaussian noise respectively. The relative errors are summarized in Table 7.2. Along with the reconstructed data seen in Figure 7.9 we can conclude that the using this starting guess, the Monte Carlo solutions are not close as close to the exact solution as when using $\boldsymbol{x}_{\text{exact}}$ as starting guess. But the reconstruction is still better than using only the deterministic solution, described in [7]. Looking at the misfit plot in Figure 7.10 we see that the burn-in period is very short even though we look at the low level of noise on data..

The idea of the reverse ray tracing algorithm is to exploit another starting guess, which is reasonable. Taking this simple test problem and using the reverse ray tracing to find a starting guess, we use this to get the reconstructions seen in Figure 7.11

So far we have used starting guesses, which are relatively close to the exact

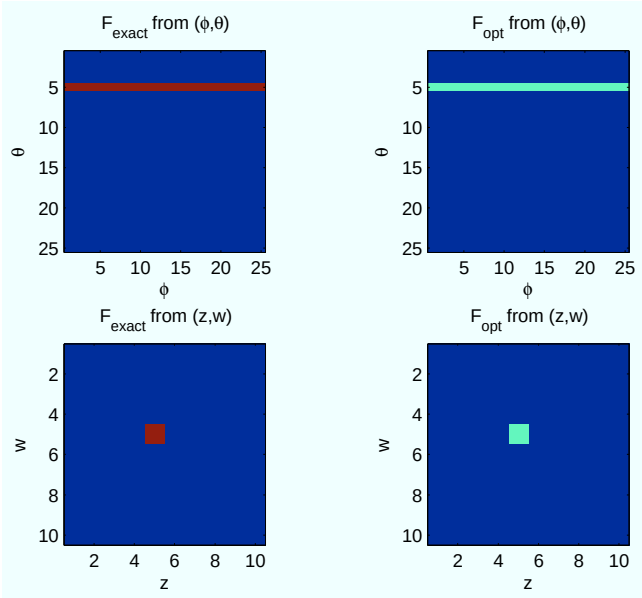

Figure 7.7: Cross sectional plot from the point (5,5) and the angle pair (5,5) using an ART solution as starting guess and 10 % Gaussian noise.

solution. Using a starting guess far away from the solution will tell us more about the robustness of the method. Therefore we choose to look at the case, where the starting guess consists of zeros. The reconstructions using this starting guess are illustrated in Figure 7.12 and 7.13. The relative errors in Table 7.3 show that using 1 % noise in data results in a smaller relative error. It is important to notice, that using zeros as starting guess, we approach the exact solution more than when using the deterministic solution to the problem as starting guess. This might be due to the exact solution, which consists of only a small number of nonzero elements. Looking at the misfit plots in Figure 7.14 we see that the burn-in period is significantly longer, than when using the

Noise level	0.1	0.01
Rel. error on solution	0.5126	0.0820
Rel. error on data	0.4498	0.0340
Step length	$1 \cdot 10^6$	$1 \cdot 10^6$

Table 7.3: The relative errors on the solution and the corresponding data based on a mean of all solutions, the corresponding step length s using a solution found by ART method as starting guess.

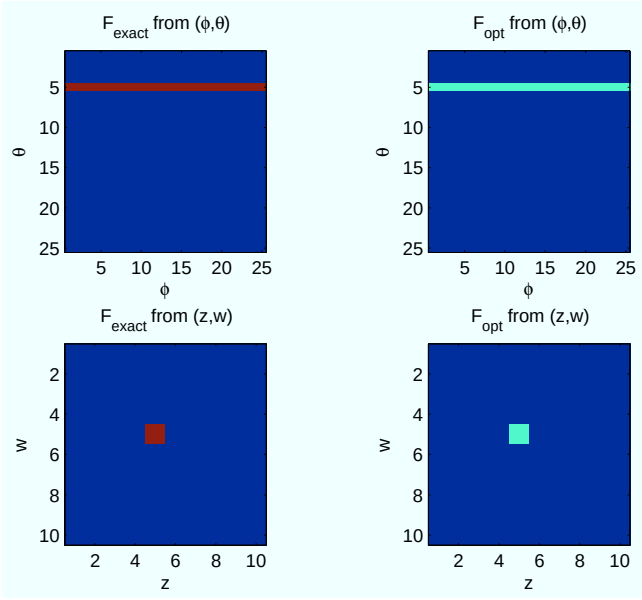

Figure 7.8: Cross sectional plot from the point (5,5) and the angle pair (5,5) using an ART solution as starting guess and 1 % Gaussian noise.

deterministic solution as starting guess. It makes sense, that when the starting guess is far away from the exact solution, the burn-in period is longer.

In Figure 7.15 a histogram of three parameters are seen. Each histogram correspond to looking at the same index on all Monte Carlo solutions. This illustrates the distribution of intensities. The corresponding three parameters from x_{exact} are 3944019, 1578531 and 3944019. We see that each histogram are centered around these three values. This also verify that looking at the mean value of all solutions make sense.

Finally we try to combine the deterministic solution and the starting guess using the reverse ray tracing algorithm. It might decrease the number of iterations in the burn-in period, which is always an issue. The reconstructions are seen in Figure 7.16 and 7.17.

All the experiments so far has been based on the prior information obtained by the reverse ray tracing algorithm. Tests are also done with a wider prior information. This time the distribution of active pixels on the source consists of a equal probability of all pixels on the source. The results from the experiments are seen in Appendix B. In general the relative errors increase a bit, and the

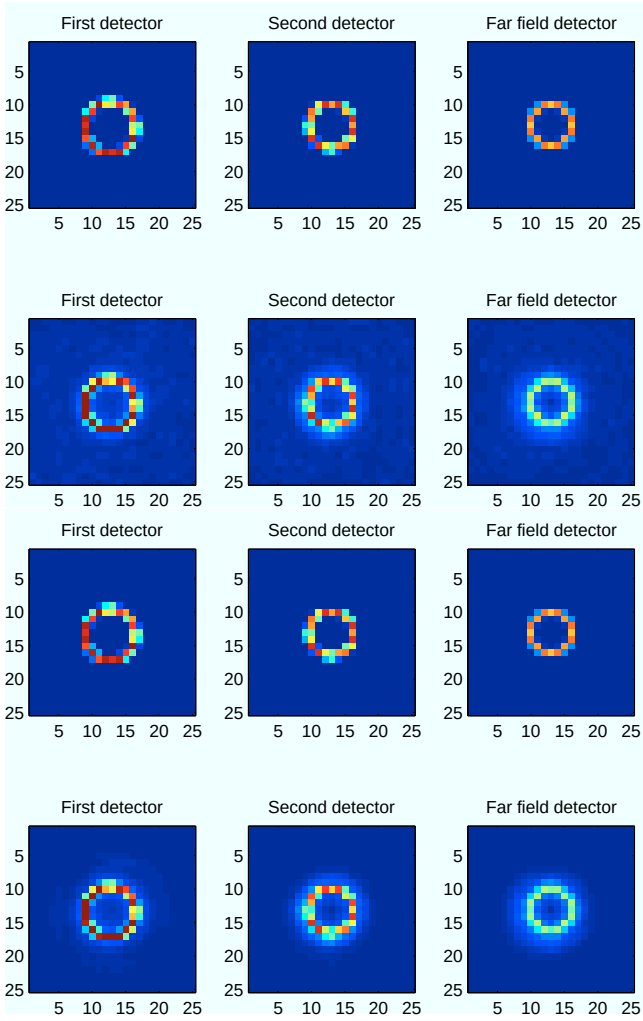

Figure 7.9: Data when using an ART solution as starting guess. Left figure correspond to 10 % Gaussian noise on data and right, 1 %.

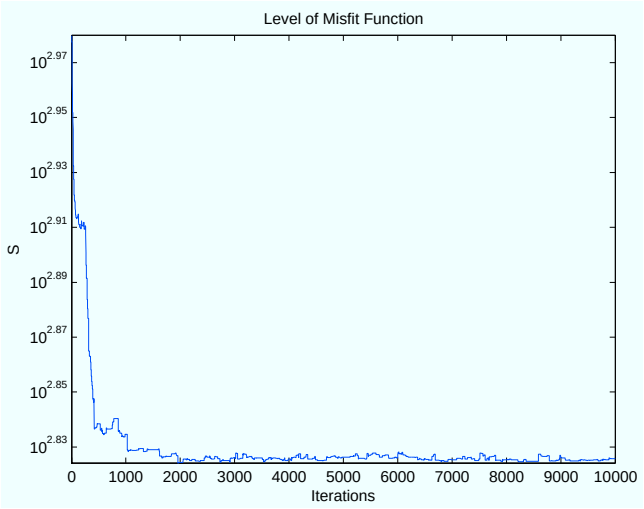

Figure 7.10: Misfit plot from the Monte Carlo inversion using zeros as starting guess and 10 % Gaussian noise in data.

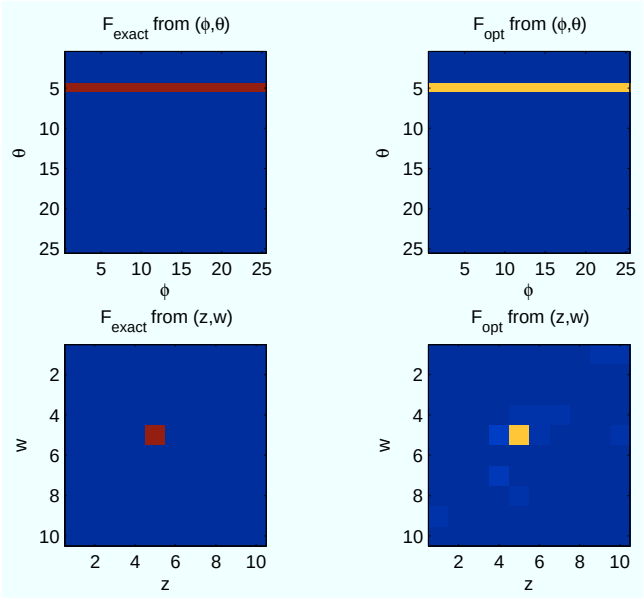

Figure 7.11: Cross sectional plot from the point (5,5) and the angle pair (5,5) using the output from the reverse ray tracing function as starting guess and 10 % Gaussian noise.

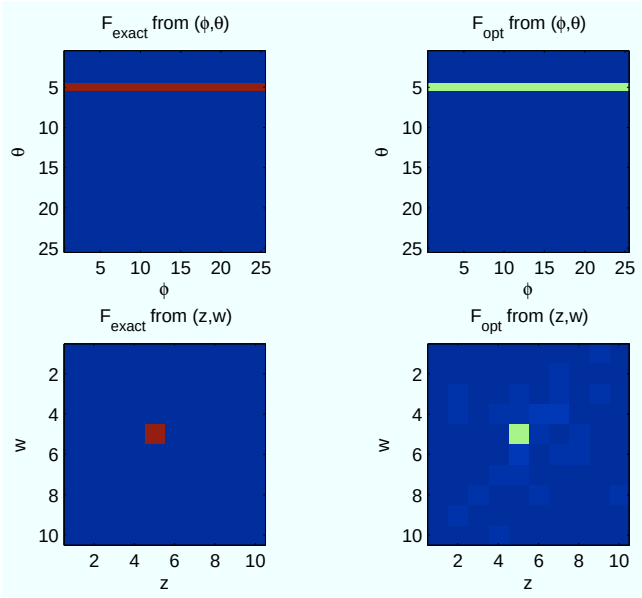

Figure 7.12: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zerosas starting guess and 10 % Gaussian noise.

burn-in periods increase as well.

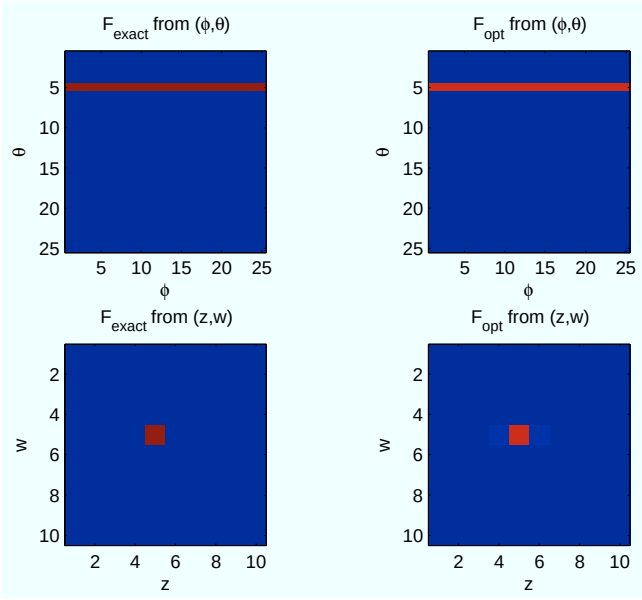

Figure 7.13: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 1 % Gaussian noise.

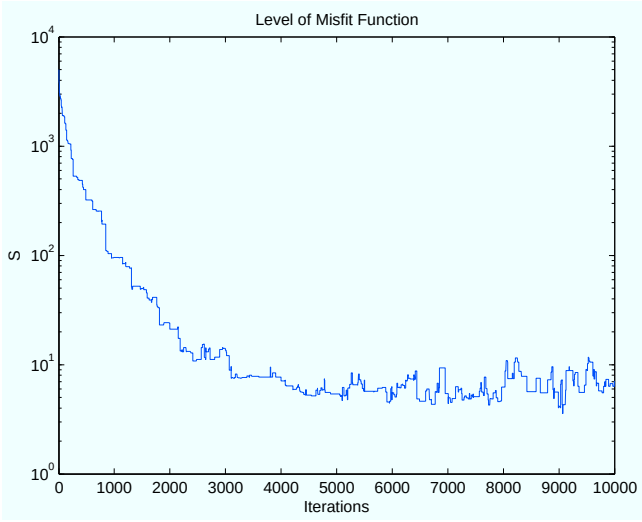

Figure 7.14: Misfit plot from the Monte Carlo inversion using zeros as starting guess and 1 % Gaussian noise in data.

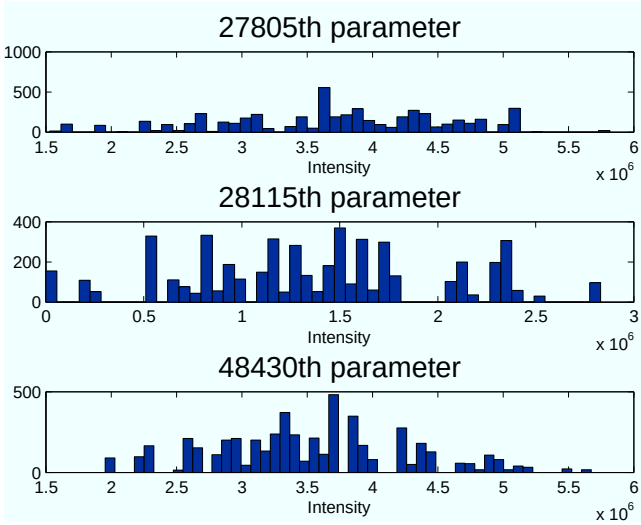

Figure 7.15: Histogram showing the distribution of three parameters of the Monte Carlo solutions obtained, when using 10 % Gaussian noise and zeros as starting guess.

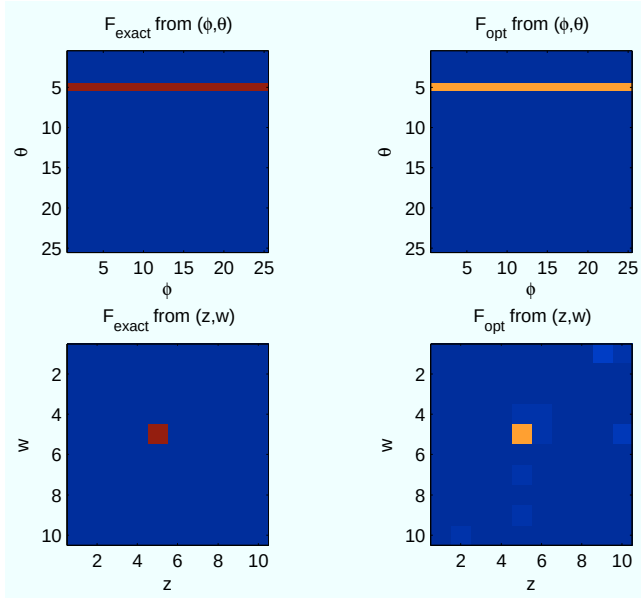

Figure 7.16: Cross sectional plot from the point (5,5) and the angle pair (5,5) using an ART solution as starting guess and 10 % Gaussian noise.

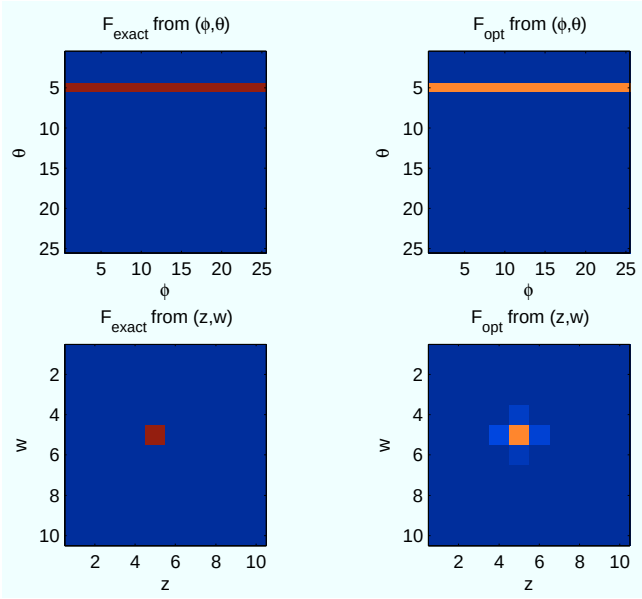

Figure 7.17: Cross sectional plot from the point (5,5) and the angle pair (5,5) using an ART solution as starting guess and 1 % Gaussian noise.

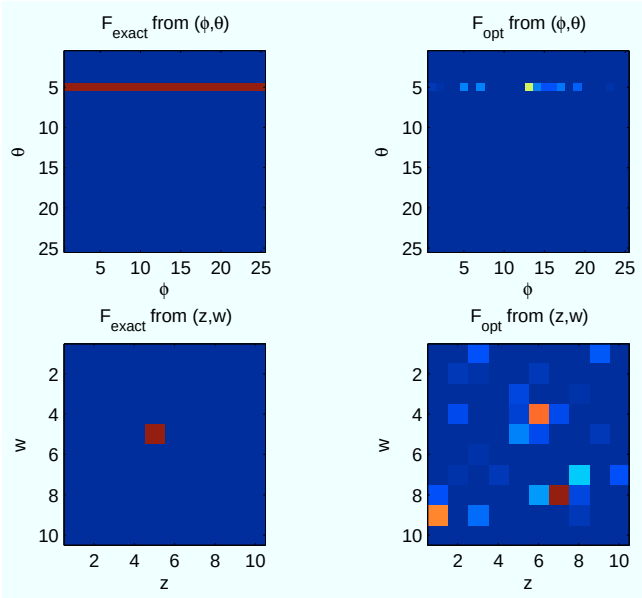

Figure 7.18: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 10 % Gaussian noise.

7.2.1 Variation along ϕ

So far we have used the prior knowledge, that there was no variation along ϕ. This is not a fully realistic assumption. It would be more reasonable to allow variation along ϕ. Most likely there will appear a kind of clustering of intensity along ϕ. First we will look at the same problem as in the previous section, but now variation along ϕ is allowed. Starting with the results using zeros as starting guess the reconstruction is seen in Figure 7.18. From a quick glance at the reconstruction it is clear to see, that the inversion method does not obtain a solution as close to the exact as in Figure 7.12. By decreasing the noise in data, the reconstruction improves, but still the reconstruction is not acceptable - see Figure 7.19.

In Appendix B it is obvious that using this prior, the method does not find the solutions close to the exact one, no matter which starting guess we use. It seems, that there are too many unknown parameters. Therefore we have to add more prior knowledge to the method. Instead of using the distribution of active source pixel obtained from the reverse ray tracing algorithm, we specify the active pixels on the source as prior information. Looking at the reconstruction in Figure 7.20, it improves significantly.

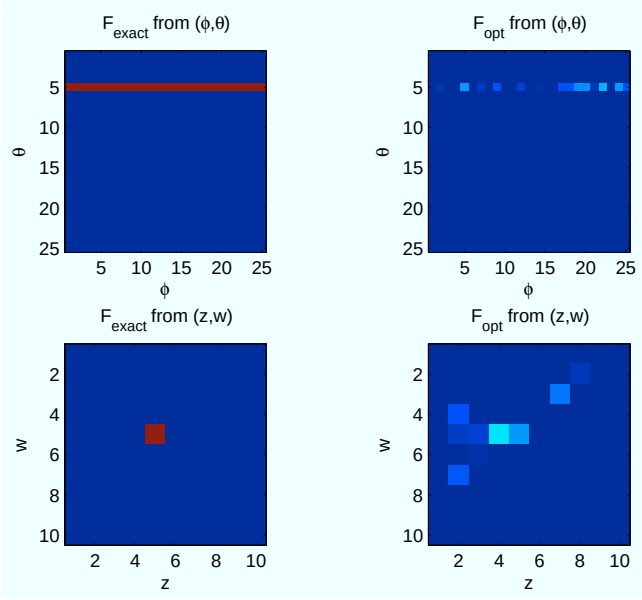

Figure 7.19: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 1 % Gaussian noise.

Using the other starting guesses the same results appear. But when using the ART solution, it seems that the realizations are not accepted, when there is 1 % noise on data - see Appendix B. When the noise level is low, the error on data, when finding the misfit value is low as well. This means that the chance of acceptance decreases. Here the error on data is so low, that no realization is accepted. As discussed earlier the Monte Carlo method performs best, when the noise on data is high, and we see that in this case.

Now we want to add a pixel on the source, a θ value and introduce variation along ϕ in the test problem as described in Section 7.1. Since the results using the deterministic solution as starting guess sometimes gives bad results we look at the results using zeros as starting guess. The reconstructions are seen in Figure 7.21 and 7.22. The reconstructions along with the errors in Table 7.4 tell us, that the method performs well. It is still based on a very small prior, which might not reflect the actual knowledge we have about the problem. Looking at the data corresponding to the reconstruction with 1 % noise in data in Figure 7.23, we see that the solutions describe the data very well.

If we for instance look at the results using the ART solution as starting guess, we see that the method perform slightly worse, see Figure 7.24 and 7.25 and the

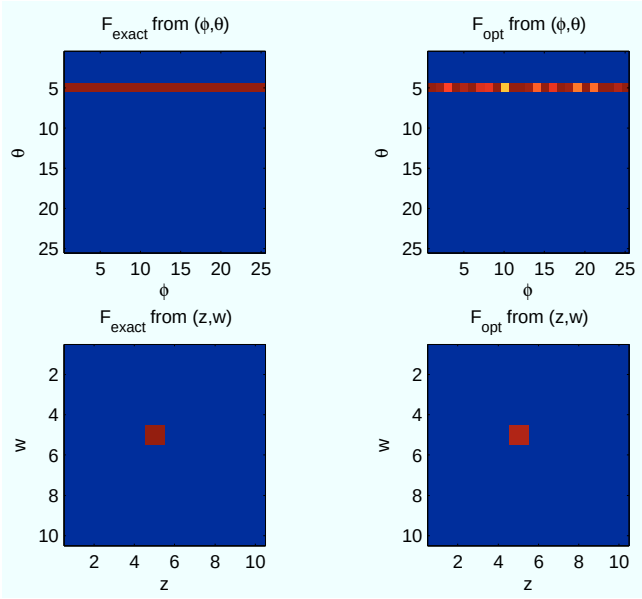

Figure 7.20: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 10 % Gaussian noise.

Noise level	0.1	0.01
Rel. error on solution	0.5761	0.1482
Rel. error on data	0.7742	0.0872
Step length	$1 \cdot 10^7$	$7{\cdot}10^5$

Table 7.4: The relative errors on the solution and the corresponding data based on a mean of all solutions, the corresponding step length s using a solution found by ART method as starting guess.

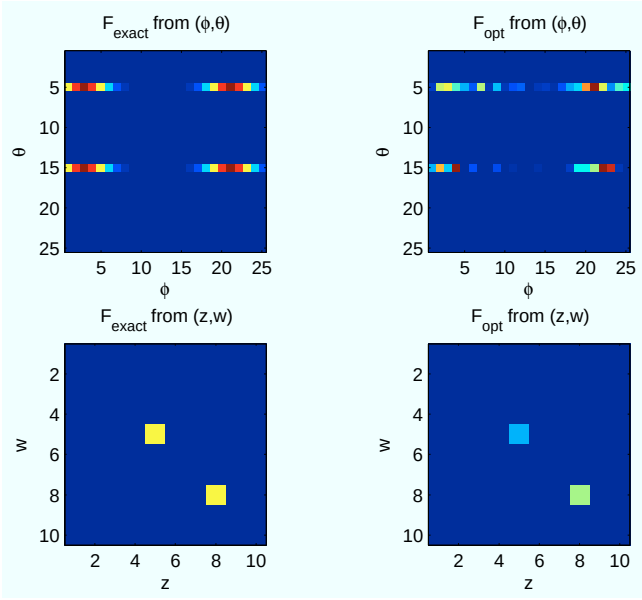

Figure 7.21: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 10 % Gaussian noise.

data in 7.26.

From the rest of the results seen in Appendix B, we can conclude, that the reconstructions with variation in ϕ are further away from the exact solutions, than the ones without variation. That is due to the increased number of parameters that the Monte Carlo method has to perturbate on. Even though the number of Monte Carlo iterations increase significantly the solutions do not come any closer to the exact solution.

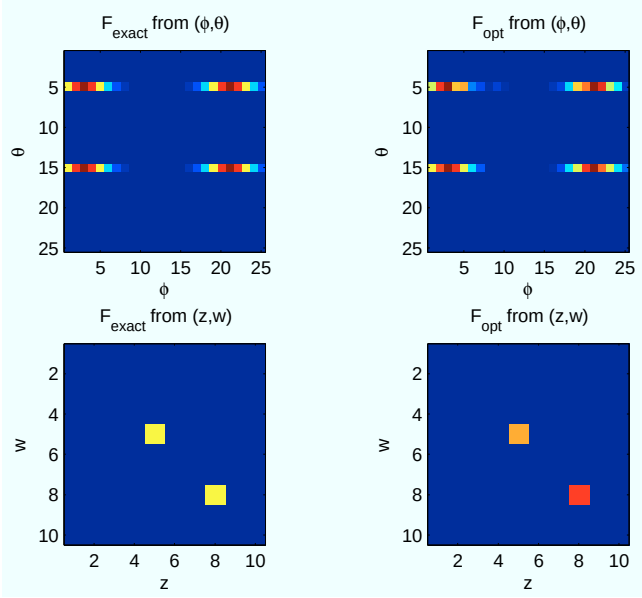

Figure 7.22: Cross sectional plot from the point (5,5) and the angle pair (5,5) using zeros as starting guess and 1 % Gaussian noise.

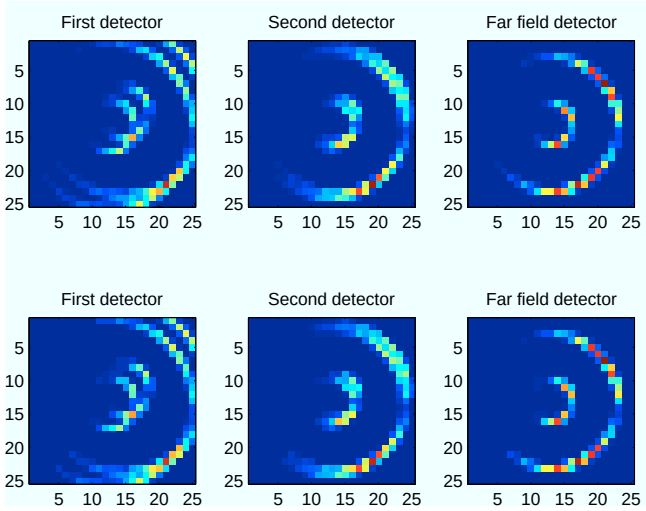

Figure 7.23: Top: exact data. Bottom: data based on the reconstruction from using zeros as starting guess and 1 % Gaussian noise.

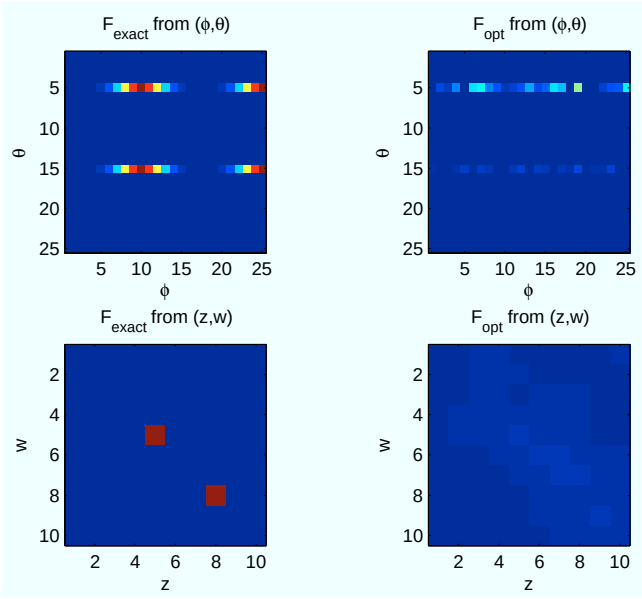

Figure 7.24: Cross sectional plot from the point (5,5) and the angle pair (5,5) using a deterministic solution as starting guess and 10 % Gaussian noise.

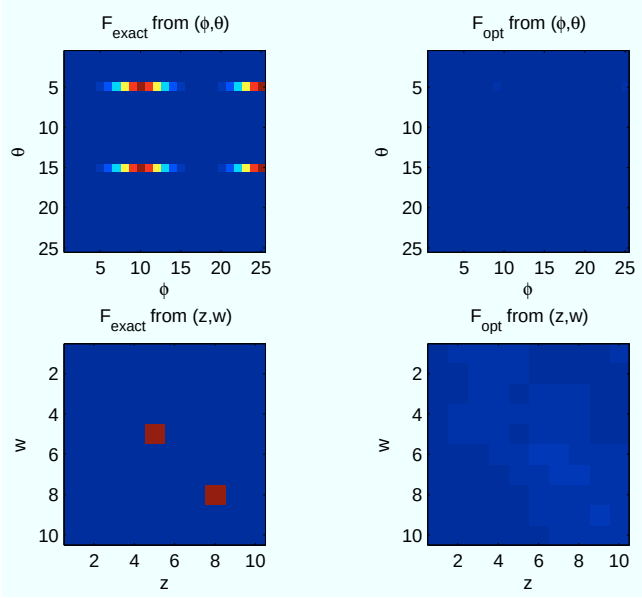

Figure 7.25: Cross sectional plot from the point (5,5) and the angle pair (5,5) using a deterministic solution as starting guess and 1 % Gaussian noise.

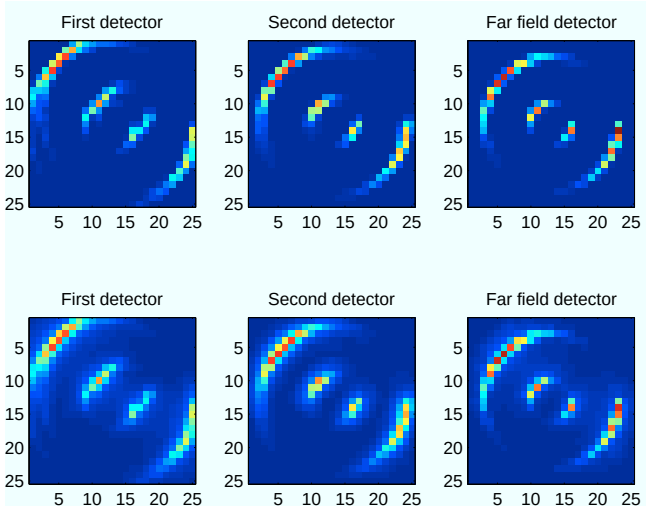

Figure 7.26: Top: exact data. Bottom: data based on the reconstruction from using a deterministic solution as starting guess and 1 % Gaussian noise.

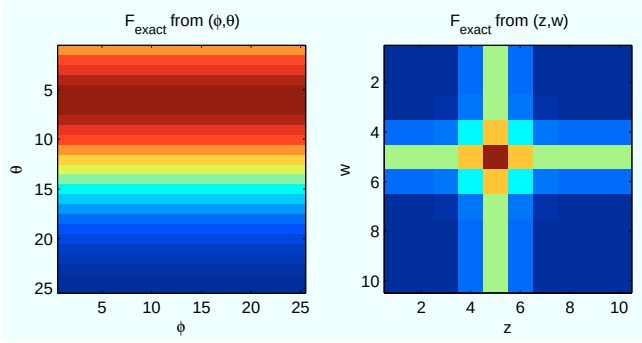

Figure 7.27: Complex problem with no variation in ϕ. Cross sectional plot seen from the point (5,5) and the angle pair (1,10)

7.3 Complex Problem

So far we have only focused on simple test problems, and now we want to extend the experiment with a more complex test problem. The test problem will describe many cones coming out from several pixels on the source. It will also be described with many θ values corresponding to several cones of light emitted from each pixel. We again start off by dealing with a test problem without variation in ϕ and it is illustrated in Figure 7.27.

To illustrate how the stochastic method deals with the more complex problem we have chosen to look at the result using $\boldsymbol{x}_{\text{exact}}$ as starting guess. Looking at the reconstruction in Figure 7.28 for 10 % noise and Figure 7.29 for 1 % noise. In both figures, a video showing every 50'th Monte Carlo solution can be found. These videos gives a clear idea about how the stochastic method works. Where the solutions changes the most is also where the largest error is present in the solution.

From this experiment it is obvious that dealing with this complex problem adding 10 % noise on data gives the methods room to manoeuvre, meaning that realizations are accepted even though they might be far away from the exact solution. Looking at the data for both values of the noise level in Figure 7.30 and 7.31 we see that the solutions corresponding to the high level of noise does not describe the data very well. Even though we use the exact solution as starting guess, it has too few restrictions, so the Monte Carlo method allows solutions far away from the exact solution.

Now we want to see how the method deals with another starting guess. Due

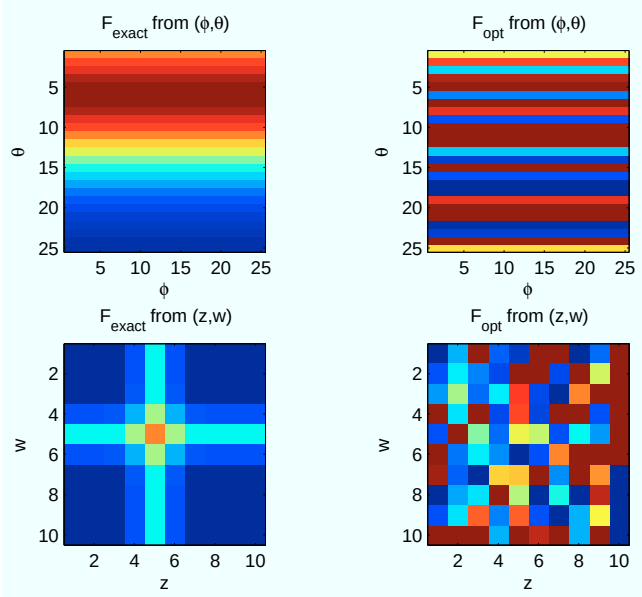

Figure 7.28: Reconstruction with no variation in ϕ and using $\boldsymbol{x}_{\text{exact}}$ as starting guess. Cross sectional plot seen from the point (5,5) and the angle pair (1,10) with 10 % Gaussian noise. Video showing some of the solutions: http://www.youtube.com/watch?v=B0gN4VVUHAk&feature=youtu.be

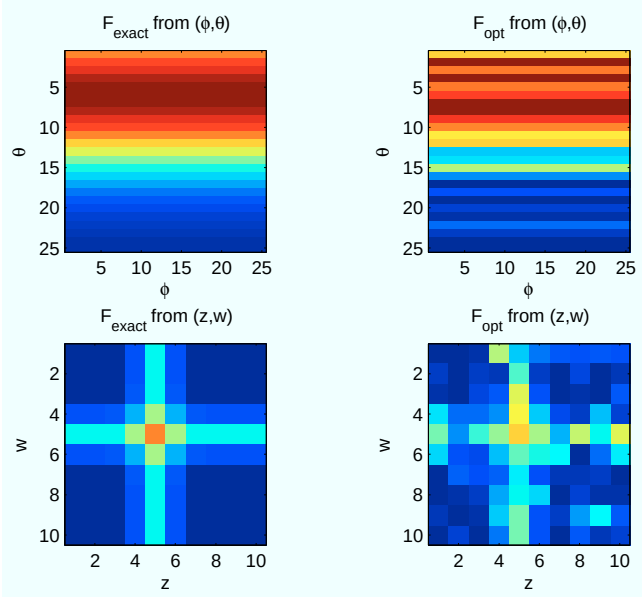

Figure 7.29: Reconstruction with no variation in ϕ and using $\boldsymbol{x}_{\text{exact}}$ as starting guess. Cross sectional plot seen from the point (5,5) and the angle pair (1,10) with 1 % Gaussian noise. Video showing some of the solutions: http://www.youtube.com/watch?v=5VHfQtxDWjc&feature=youtu.be

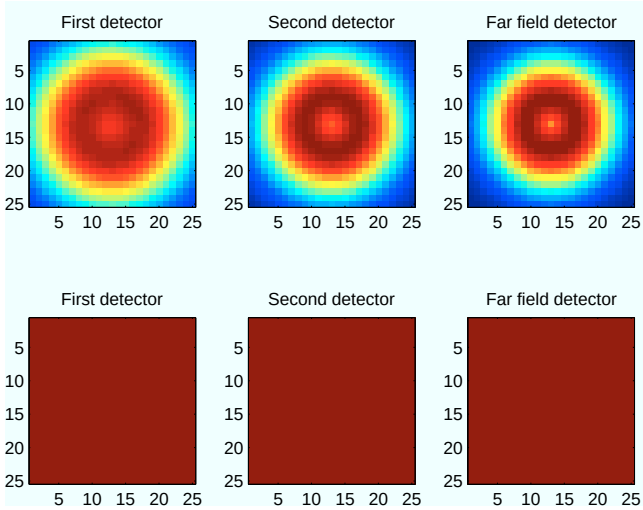

Figure 7.30: Data with no variation in ϕ and 10 % Gaussian noise and using $\boldsymbol{x}_{\text{exact}}$ as starting guess. Video showing some of the corresponding data solutions: http://www.youtube.com/watch?v=4C7ZrtCBGTc&feature=youtu.be

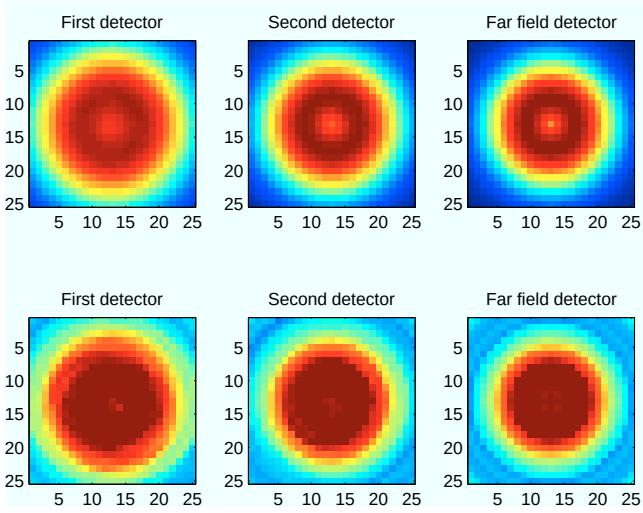

Figure 7.31: Data with no variation in ϕ and 1 % Gaussian noise and using $\boldsymbol{x}_{\text{exact}}$ as starting guess. Video showing some of the corresponding data solutions: http://www.youtube.com/watch?v=ueu35HTdHsw&feature=youtu.be

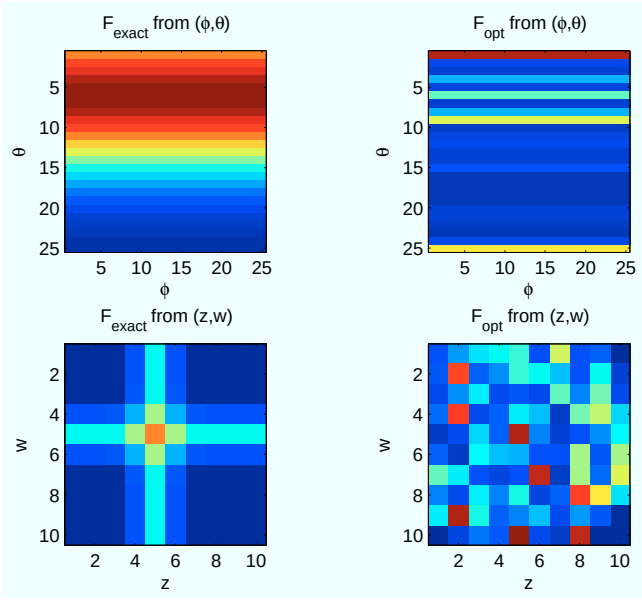

Figure 7.32: Reconstruction with no variation in ϕ, with 1 % Gaussian noise and using zeros as starting guess. Cross sectional plot seen from the point (5,5) and the angle pair (1,10). Video showing some of the solutions: http://www.youtube.com/watch?v=SfMa4GyPXCc&feature=youtu.be

to the results above, we will only focus on the case, where 1 % noise is added. We will start off by choosing the starting guess consisting of zeros. The corresponding reconstruction and data is seen in Figure 7.32 and 7.33 along with the videos showing the solutions. From the relative errors in Table 7.5 we see, that even though the error in the solution is large, they are still describing the data almost equally as well, when using $\boldsymbol{x}_{\text{exact}}$ as starting guess. From the video, we can see that the point (5,5) at the source in the reconstructions, is always of high intensity. That pixel is affecting the corresponding data in a way, such that the data does not differ much from the true data. So even though each solution is far away from the exact solution, it still might results in a small residual in the stochastic method.

Summing up on the results we have seen that the method seems to be able to find reasonable solutions. The performance of the method has shown to be very affected by the prior information given to the method. When more restrictions were imposed on the method, the outcome were closer to the exact solution. In this chapter we focused at two levels of noise in data. It did turn out, that with 1 % noise on data the Monte Carlo method did perform well. As mentioned in

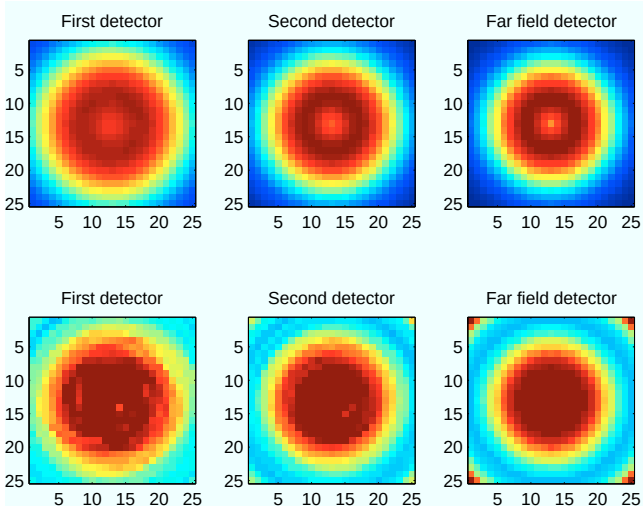

Figure 7.33: Data with no variation in ϕ, with 1 % Gaussian noise and using zeros as starting guess. Video showing some of the corresponding data solutions: http://www.youtube.com/watch?v=3TlqUKH3oKA&feature=youtu.be

Noise level	Rel. error on solution	Rel. error on data	Step length
0.01	1.1629	0.2571	10^6

Table 7.5: The relative errors on the solution and the corresponding data based on a mean of all solutions, the corresponding step length s using a solution found using zeros as starting guess.

Section 5 the sampling method prefers high level of noise.

We also saw that the method was able to solve even the more complex problems. Especially when there were no variation along ϕ, the method was able to locate the pair of angles. When it comes to the spatial variation, the method allowed more variation far from the exact solution. This is due to the properties about the far-field detector - also described in [7].

Chapter 8

Conclusion

The aim of this thesis was to understand the mathematics behind a physical experiment and use that knowledge to construct a problem, which was solved using a stochastic method. Also experimenting with a hybrid of both a stochastic and a deterministic method was an important aspect of the thesis. The hybrid part was basically that a deterministic solution to the inverse problem was a good starting point for the stochastic method.

We found that the stochastic method was able to solve the different simple test problems dependent on noise level, starting guess and prior. The prior information given to the model had a great influence on the performance of the method. The Monte Carlo method thrives with few restrictions, but when adding many restrictions it approaches the exact solution.

When a complex problem was considered it was easier to see, how the method actually performed. From the videos it was shown, where the intensities varied the most and where they seemed constant. When looking at the solutions the angular variation was much better described than the spatial one. As mentioned in the previous chapter, that is due to presence of the far-field detector. Basically it does not relate to the actual stochastic method, but to the physical set-up.

The method implemented turned out to be quite robust, it was able to find solutions describing the data from all kinds of starting guesses. Even though

zeros were used as starting guess it found solutions describing the data. The burn-in period was affected by the choice of starting guess, the noise level and the complexity of the problem. We saw that when using a deterministic solution to the problem as starting guess the burn-in period was relatively short compared to using zeros.

We have seen that the Monte Carlo method can be used to solve this inverse problem based on a diffraction problem. We have analyzed the results and the performance of the stochastic method. We have also shown that the deterministic solution can be used to decrease the burn-in period.

8.1 Future Work

Working in MATLAB we often experience limitations in capacity and computation availability, when working with large-scale. The test problems used in the four dimensional problem were not full scale. The grid on the source could be refined, which would increase the dimensions of problem. Then the computations would be impossible to perform in MATLAB due to lack of computation capacity. One of the obvious steps to take after this thesis is to implement the algorithm in another language. It is also a possibility of combining the actual implemented MATLAB code with code implemented in C with using the mex-compiler in C. This was also an issue in this thesis, but due to limitations of time, it was skipped.

Focusing on the actual stochastic method many improvements could be considered. In the part concerning the prior information other aspects could be considered. It might be possible to include more knowledge about the actual experiment. Also it would might improve the method to play with other accept criteria. For instance the step length was fixed, but it might improve the performance of the method if the step length was adaptable. It could also be a possibility to use other stochastic methods. Investigations could reveal if they are more suitable for this specific problem setting.

This physical problem has been treated as a general ray tracing problem, and the model described in Chapter 3 was based on that. It has shown that this model has some weaknesses, mainly that the spatial variation was hard to detect. Therefore it might increase the efficiency of both a deterministic and a stochastic solution if another model was chosen to describe the problem. A model that was able to handle the spatial variation better.

Appendix A

Introductory Investigations

To acquire some understanding about how the Monte Carlo inversion deals with a problem, which is based on materials science, some simple tests are done. The experiments consist of several test images with blobs of varying sizes and intensities. The aim is now to find the resolution limit for each test image, that is to find for instance the maximal distance between blobs, where the reconstruction is still possible. The tests are made with the following images, see Figure A.1

To investigate if the distance between two blobs plays a role, when reconstructing

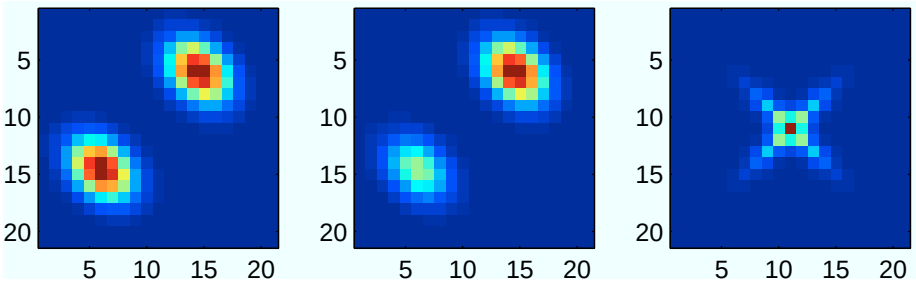

Figure A.1: 4 different test images used to find resolution limits.

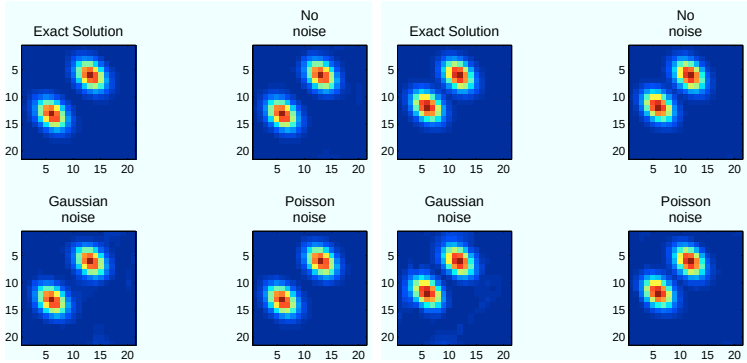

Figure A.2: First test image with two distances with 1 % Gaussian noise and 0.2 % Poisson.

we have looked at the image corresponding to the first image in Figure A.1 with two equally sized blobs. In real tests we will not have the exact data, so therefore we are test with data containing noise, that is Gaussian distributed noise and Poisson noise. This test will also be able to give an indication about how robust the method is. If it is very sensitive so small changes in the noise level the method is not that robust.

The algorithm it self is based on the simple implementation of a Monte Carlo method described in Section 5.3. It uses a SVD transformation to transform $\boldsymbol{A}$ and $\boldsymbol{b}$ into the right domain, so the Monte Carlo method is able to solve the problem. Therefore it is of course necessary to make a corresponding transform to the output of the algorithm. Also since the system is underdetermined, it is necessary to add a null space to the solution. The ensures that the solution is in the right solution space.

Solving this problem using only a small noise level, that is 0.01 % for the Gaussian noise and 0.002 % for the Poisson noise a reconstruction is generated in Figure A.2. We see that the reconstructions are good, but again the noise level is also very low. We therefore in Figure A.3 start with changing the level of Poisson noise to 0.01 % and see that the noise is more present. Adding more noise, we see that at the solutions start to be dominated by noise - see Figure A.4 and Figure A.5. It seems that the image with Gaussian distributed noise makes better reconstructions. Thinking about how the method works, it makes sense that it handles Gaussian noise better than Poisson noise. The perturbations in the method actually is based on a Gaussian distribution.

Looking at the second test image with two blobs of different size, we see some of the same results - in Figure A.5. From the figure we can conclude that

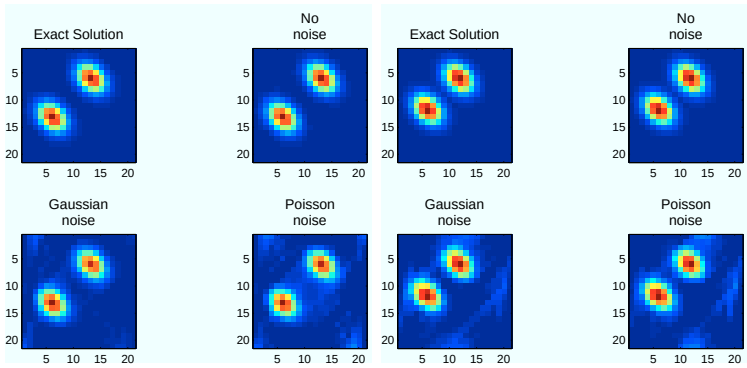

Figure A.3: First test image with two distances with 1 % Gaussian and Poisson.

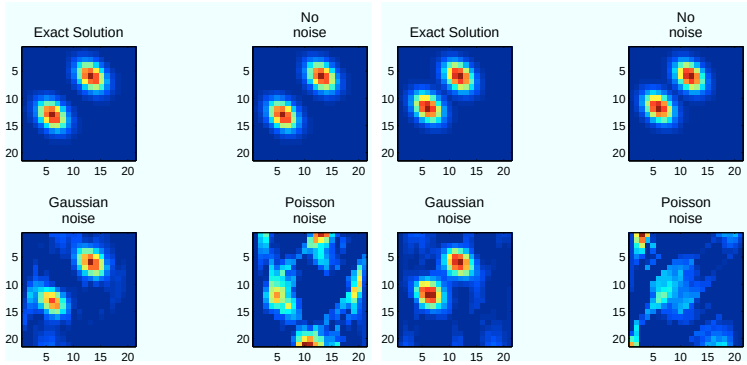

Figure A.4: First test image with two distanceswith 5% Gaussian and Poisson.

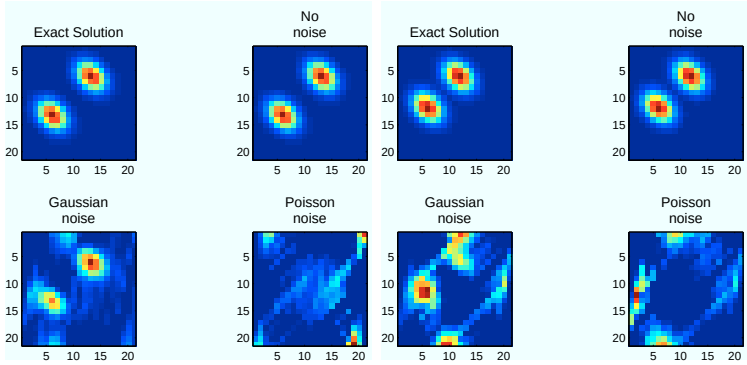

Figure A.5: First test image with two distances with 10% Gaussian and Poisson.

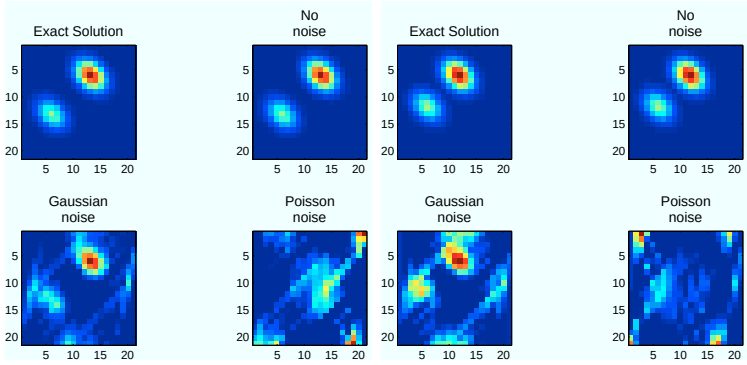

Figure A.6: Second test image with two distances with 10% Gaussian and Poisson.

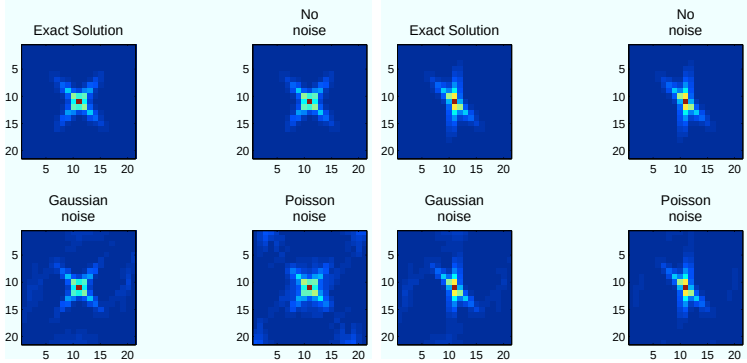

Figure A.7: Third test image with two different angles with 1% Gaussian and Poisson.

the smaller the distance the worse the reconstruction or you can say the more affected by noise is the reconstruction. This is quite intuitive, but what happens when the two blobs are connected? The experiments with the third image can say something about that.

In Figure A.7 we see the perfect reconstruction of two slim blobs intersecting. But again if more noise is added first the reconstruction based on data with Poisson noise is influenced by noise and then the other reconstruction is affected, that is when the noise level is 10 % - see Figure A.8 and A.9.

So far we have looked at more or less Gaussian bells, but it could be interesting to see how the method works with a totally different shape. Therefore some tests with a figure shaped like a triangle is carried out. The triangle has the same

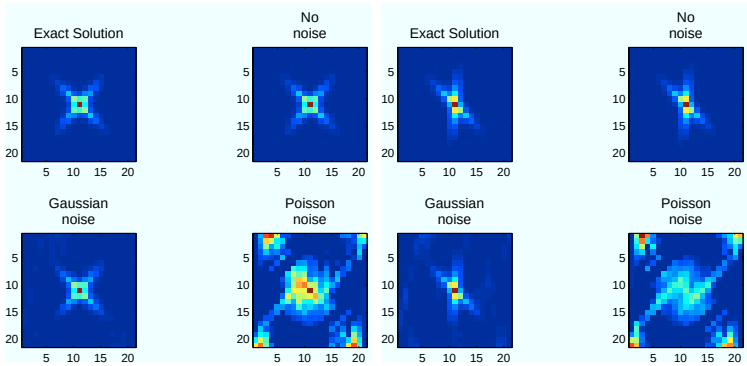

Figure A.8: Third test image with two different angles with 5% Gaussian and Poisson.

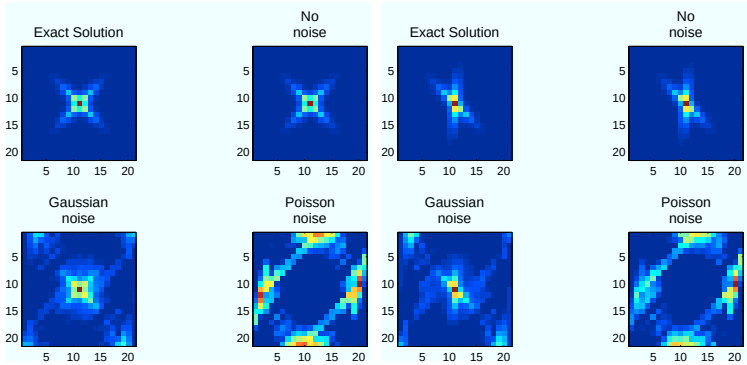

Figure A.9: Third test image with two different angles with 10% Gaussian and Poisson.

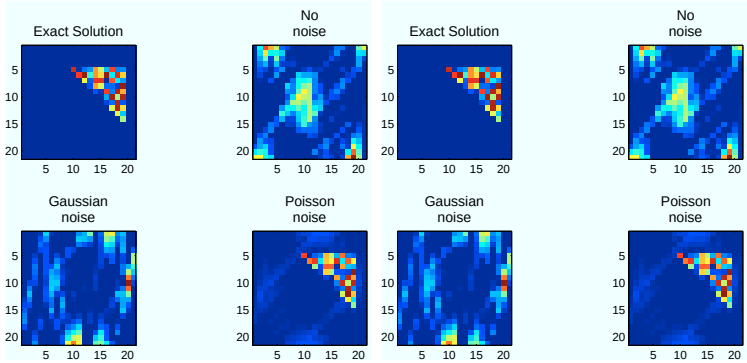

Figure A.10: Testing with a trianglular shape with 1% Gaussian and Poisson and right with 5 %.

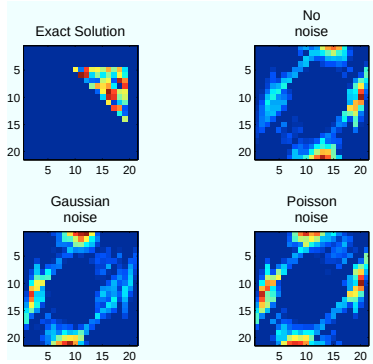

Figure A.11: Testing with a triangular shape with 10% Gaussian and Poisson.

intensity as the previous test examples. We would just like to know, whether this method only is able so solve problem based on Gaussian shapes. As above we start off by using a small noise level, 1 % noise, and the reconstructions are shown in Figure A.10 and Figure A.11. Now this figure shows that actually the reconstructions with the data containing Poisson noise performs best. This is a contradiction from what we saw with the other shapes. But on the other hand, this is not formed as a Gaussian bell, and therefore containing Gaussian noise might be a problem. But again the reconstruction without noise should not perform worse than with Poisson noise. It is probably due to the constant we use for normalization, when we construct the right noise level for the Poisson noise, described in Section 3.3.

From the experiments above we found that the method seems very sensitive to

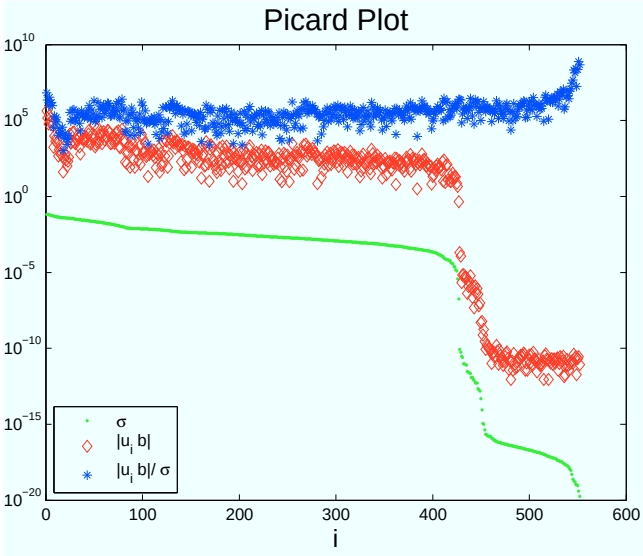

Figure A.12: Picard plot representing the singular values and the SVD coefficients of the matrix $\boldsymbol{A}$.

the level of noise on the data, and also type of noise. One of the problems with the simple implementation is that the problem is underdetermined, so therefore we have to take into consideration that our solution is not in the right space and therefore has to be converted. For simplicity we now introduce a new system of linear equation $\boldsymbol{Ax} = \boldsymbol{b}$, which is a quadratic problem, so $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{R}^n$. in this way we can still perform the SVD transformation, but then the transformed output will be in the correct solution space. We will perform some of the same experiments as described in the appendix and then afterwards we will look different types of a priori. The SVD transform is trivial, since the problem is quadratic. We start off by making a so-called Picard plot, which tells us, where the Discrete Picard condition is satisfied, e.q. where the problem is solvable. Looking at the Picard plot in Figure A we see that the singular values as expected is a descending sequence. The SVD coefficients $|\boldsymbol{u}_i^T \boldsymbol{b}|$ are faster decaying than the singular values until some index τ. After this point the fraction $|\boldsymbol{u}_i^T \boldsymbol{b}|/\sigma_i$ starts to increase and the singular values drop to the machine precision. This means that the solution will be dominated by larger SVD coefficients corresponding to smaller singular values and this will lead to a solution that is dominated by noise and will tend to the naive solution.

We now conclude that we should not include SVD coefficients, which correspond to an index larger than $i = 420$. The investigation will deal with determing the

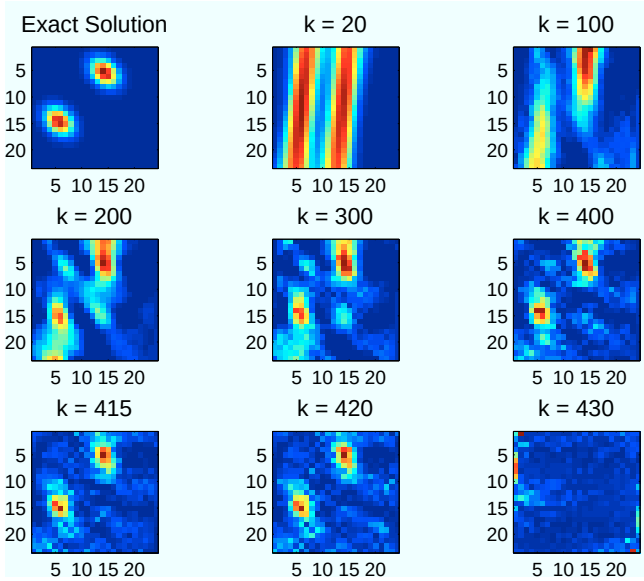

Figure A.13: Picard plot representing the singular values and the SVD coefficients of the matrix $\boldsymbol{A}$.

k	20	100	200	300	400	415	420	430
Error	0.861	0.772	0.722	0.659	0.583	0.613	0.687	$3.013 \cdot 10^5$

Table A.1: The relative errors on the solution and their corresponding index k.

optimal index k, which represent the index, where all SVD components should be included until, to obtain the best reconstruction. We now solve the problem with different values of k to investigate, which k should be the optimal. In Figure A.13 the reconstructions are seen, which is a mean of all the possible solution after the burn-in period, and in Table A.1 the relative errors are computed. We see that if k is too small the reconstruction is very dependent on the right singular vectors, which also makes sense. The larger k gets, the better reconstruction, but as k tend to around 400 it seems that the reconstructions are not that smooth, but more grainy.

It is also important to notice that when $k > 420$ the reconstruction is totally dominated by noise. From the reconstructions in the figure it can be hard to tell, which reconstruction is best, therefore we focus on the relative error of the solution. It seems that the optimal k is $k = 400$. In this experiment we used data without noise, which is not realistic at all. Therefore we now do the same experiments with data containing 5 % Gaussian distributed noise. We still use

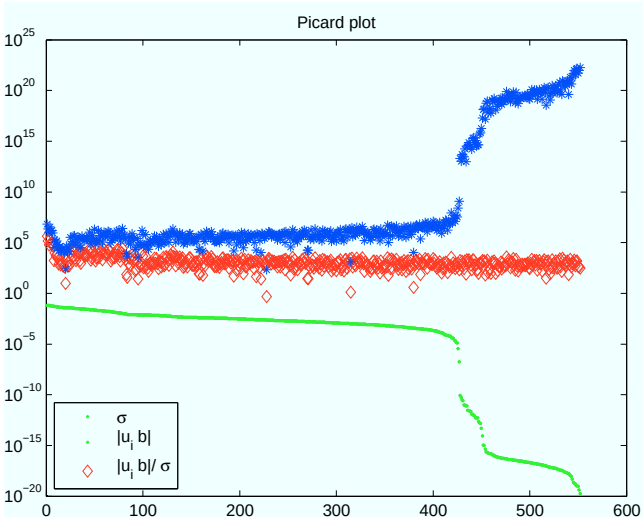

Figure A.14: Picard plot representing the singular values and the SVD coefficients including 5 % Gaussian noise.

k	20	100	200	300	400	415	420	430
Error	0.861	0.774	0.736	0.747	1.450	1.834	3.864	$2.329 \cdot 10^6$

Table A.2: The relative errors on the solution and their corresponding index k.

the exact solution as starting guess. In Figure

We know that a realistic type of noise is Poisson distributed noise, so therefore it is evident to test the method with Poisson distributed noise. We start by choosing a small noise level, $e = 0.01$ and zeros as starting guess. Looking at the reconstructions in Figure A.17 we conclude that even though the noise level is very low, the reconstructions are dominated by noise even though only a few number of the SVD coefficients are included.

This is also verified by the Picard plot in Figure A.16, where the fraction $|\boldsymbol{u}_i^T \boldsymbol{b}|/\sigma_i$ starts increasing around index 200. From the Misfit values in Figure A.18 we see that, when we use zeros as starting guess the burn-in period is much longer than when we use x_{exact}, and therefore when we found the mean of all solutions, we cannot include the first 9000 solutions.

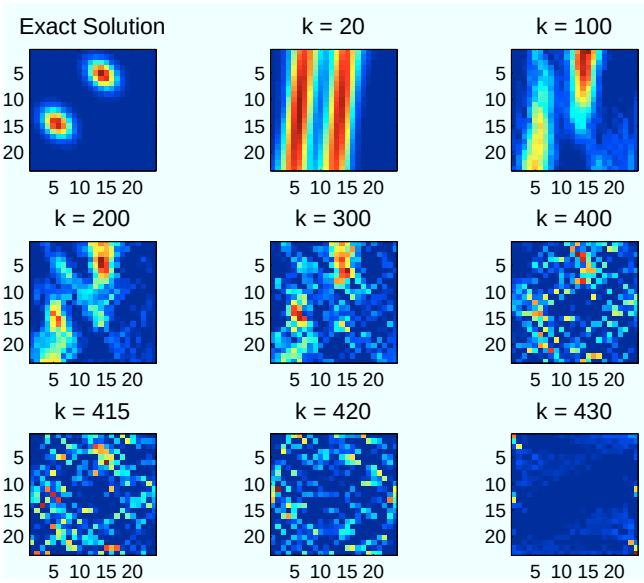

Figure A.15: The exact solution plotted along with several reconstructions using data containing 5 % Gaussian noise.

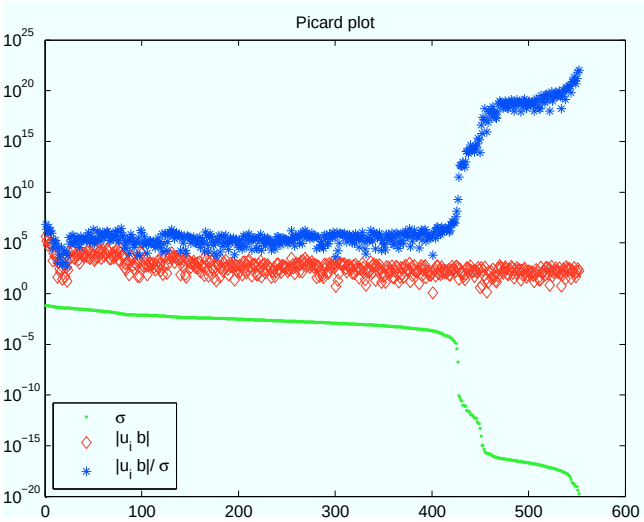

Figure A.16: Picard plot representing the singular values and the SVD coefficients including 1 % Poisson noise.

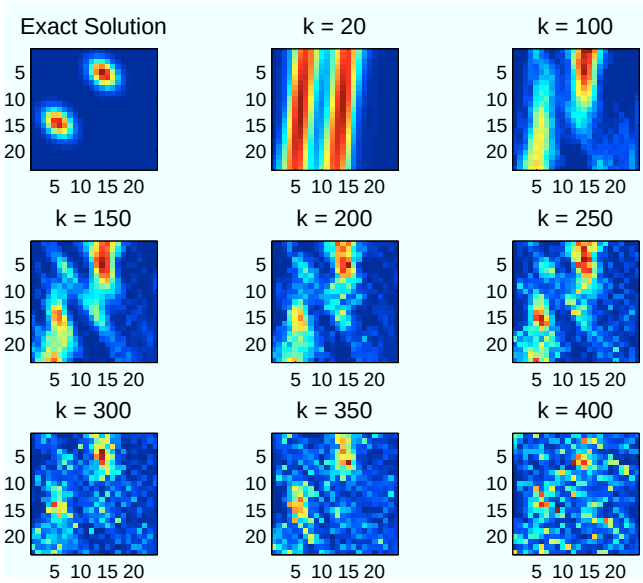

Figure A.17: The exact solution plotted along with several reconstructions using data containing 1 % Poisson noise.

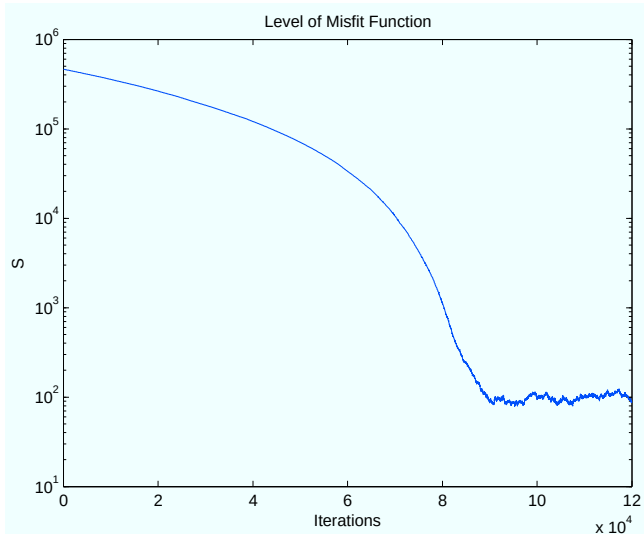

Figure A.18: The Misfit function corresponding to data containing 1 % Poisson noise.

k	20	100	200	300	400	415	420	430
Error	0.861	0.772	0.736	0.724	0.718	0.673	0.698	0.8491

Table A.3: The relative errors on the solution and their corresponding index k.

Appendix B

Results 4D

Starting Guess		Test Problem 1, No variation in ϕ			
		Ray Prior		Wide Prior	
		10 % noise	1 % noise	10 % noise	1 % noise
$\boldsymbol{x}_{\text{exact}}$	s	$5 \cdot 10^6$	$5 \cdot 10^6$	$5 \cdot 10^6$	$5 \cdot 10^6$
	rel. error on solution	0.0896	0.0072	0.2842	0.0023
	rel. error on data	0.0866	0.0049	0.2679	0.0060
	acceptance rate in %	54	50	52	50
$\boldsymbol{x}_{\text{ART}}$	s	$5 \cdot 10^7$	$1 \cdot 10^6$	$5 \cdot 10^7$	$1 \cdot 10^6$
	rel. error on solution	0.5585	0.5790	0.6042	0.6724
	rel. error on data	0.4020	0.3657	0.4108	0.3697
	acceptance rate in %	51	50	52	50
$\boldsymbol{x}_{\text{ray}}$	s	$3 \cdot 10^6$	$1 \cdot 10^6$	$3 \cdot 10^6$	$1 \cdot 10^6$
	rel. error on solution	0.3387	0.2843	0.3661	0.1783
	rel. error on data	0.2808	0.0811	0.2656	0.0594
	acceptance rate in %	54	48	55	48
$\boldsymbol{x}_{\text{zero}}$	s	$1 \cdot 10^6$	$1 \cdot 10^6$	$1 \cdot 10^6$	$1 \cdot 10^6$
	rel. error on solution	0.5126	0.0820	0.7994	0.1737
	rel. error on data	0.4498	0.0340	0.5412	0.0651
	acceptance rate in %	58	50	58	49
$\boldsymbol{x}_{\text{combi}}$	s	$3 \cdot 10^6$	$1 \cdot 10^6$	$3 \cdot 10^6$	$1 \cdot 10^6$
	rel. error on solution	0.2956	0.2746	0.3879	0.3367
	rel. error on data	0.2938	0.0796	0.2896	0.0947
	acceptance rate in %	54	46	55	48

Starting Guess		Test Problem 1, variation in ϕ			
		Ray Prior		Small Prior	
		10 % noise	1 % noise	10 % noise	1 % noise
$\boldsymbol{x}_{\text{exact}}$	s	$3 \cdot 10^6$	$3 \cdot 10^6$	$3 \cdot 10^6$	$1 \cdot 10^6$
	rel. error on solution	0.0216	0.0008	0.0288	0.0013
	rel. error on data	0.0704	0.0019	0.0844	0.0024
	acceptance rate in %	50	49	50	50
$\boldsymbol{x}_{\text{ART}}$	s	$5 \cdot 10^6$	$1 \cdot 10^6$	$1 \cdot 10^7$	$1 \cdot 10^6$
	rel. error on solution	1.1058	0.9365	0.5743	0
	rel. error on data	1.1595	0.3713	0.4924	0
	acceptance rate in %	59	55	66	0
$\boldsymbol{x}_{\text{ray}}$	s	$5 \cdot 10^6$	$1 \cdot 10^5$	$5 \cdot 10^6$	$7 \cdot 10^4$
	rel. error on solution	01.4835	0.9738	0.1907	0.1954
	rel. error on data	1.6812	0.6007	0.4494	0.1309
	acceptance rate in %	61	53	73	78
$\boldsymbol{x}_{\text{zero}}$	s	$5 \cdot 10^6$	$1 \cdot 10^6$	$5 \cdot 10^6$	$7 \cdot 10^4$
	rel. error on solution	1.4741	1.0244	0.1800	0.0970
	rel. error on data	1.7986	0.2653	0.4507	0.0935
	acceptance rate in %	60	58	71	76
$\boldsymbol{x}_{\text{combi}}$	s	$1 \cdot 10^7$	$1 \cdot 10^5$	$1 \cdot 10^7$	$1 \cdot 10^6$
	rel. error on solution	1.5052	0.9753	0.4142	0.4004
	rel. error on data	1.3654	0.5467	0.5078	0.2248
	acceptance rate in %	57	55	64	40

Starting Guess		Test Problem 2, variation in ϕ	
		Small Prior	
		10 % noise	1 % noise
$\boldsymbol{x}_{\text{ART}}$	s	$5 \cdot 10^6$	$5 \cdot 10^5$
	rel. error on solution	0.9270	1.0039
	rel. error on data	0.5559	0.3766
	acceptance rate in %	56	54
$\boldsymbol{x}_{\text{ray}}$	s	$5 \cdot 10^6$	$5 \cdot 10^5$
	rel. error on solution	0.8021	0.5119
	rel. error on data	0.7301	0.3827
	acceptance rate in %	56	42
$\boldsymbol{x}_{\text{zero}}$	s	$5 \cdot 10^6$	$5 \cdot 10^5$
	rel. error on solution	0.6091	0.0828
	rel. error on data	0.6742	0.0850
	acceptance rate in %	56	42
$\boldsymbol{x}_{\text{combi}}$	s	$5 \cdot 10^6$	$5 \cdot 10^5$
	rel. error on solution	0.5355	0.2858
	rel. error on data	0.6484	0.1987
	acceptance rate in %	56	42

Starting Guess		Complex Problem	
		No Variation in ϕ	
		10 % noise	1 % noise
$\boldsymbol{x}_{\text{exact}}$	s	$1 \cdot 10^7$	$1 \cdot 10^6$
	rel. error on solution	3.6300	0.4668
	rel. error on data	2.3877	0.1648
	acceptance rate in %	86	89
		1 % noise	0.1 % noise
$\boldsymbol{x}_{\text{ART}}$	s	$5 \cdot 10^7$	$5 \cdot 10^7$
	rel. error on solution	7.0681	5.9828
	rel. error on data	0.6767	0.0718
	acceptance rate in %	67	57
		1 % noise	0.1 % noise
$\boldsymbol{x}_{\text{zero}}$	s	$1 \cdot 10^6$	$8 \cdot 10^7$
	rel. error on solution	1.1629	6.1143
	rel. error on data	0.2571	0.3207
	acceptance rate in %	86	65

Appendix C

Matlab Code

```
for k = 1:length(Nd);
    d_tmp = d(k); h_tmp = hy(k); y_tmp = a(k):h_tmp:b(k);
    [Y, W] = meshgrid(y_tmp, w);
    theta_int = atan((Y-W)/d_tmp);
    theta_ind = ceil((theta_int+theta_max)/(2*theta_max)*Ntheta);
    Ind = 0;
    for p = 1:Ntheta
        for q = 1:Nw
            F = zeros(Nw, Ntheta);
            F(q,p) = 1; Ind = Ind +1;
            for j = 1:Nd(k)
                s = 0;
                for i = 1:Nw
                    theta1 = theta_ind(i,j); theta2 = theta_ind(i,j+1);
                    t = ones((theta2-theta1)+1,1);
                    if length(t) > 1
                        t(1)=(theta_pixels(theta1+1)...
                            -theta_int(i,j))/htheta;
                        t(end) = (theta_int(i,j+1)- ...
                            theta_pixels(theta2))/htheta;
                    else
                        t = abs(theta_int(i,j+1)-theta_int(i,j))/htheta;
                    end
                    I = htheta*(F(i,theta1:theta2)*t);
                    s = s + I;
                end
                G(j,k) = s;
            end
            Asmall(1:length(G),Ind,k) = G(:,k);
        end
    end
end
```

Bibliography

[1] P. C. Hansen. *Discrete Inverse Problems - Insight and Algorithms.* SIAM, 2010.

[2] K. Hastings, W. Monte carlo sampling methods using markov chain and their applications. *Biometrika*, (57):97–109, 1970.

[3] Klaus Mosegaard and Albert Tarantola. Monte carlo sampling of solutions to inverse problems. *Geophysical Research*, 100(B7):12,431–12,447, 1995.

[4] D.P. O'leary P. C. Hansen, J.G. Nagy. *Deblurring Images - Matrices, Spectra and Filtering.* SIAM, 2008.

[5] H.F. Poulsen H.O. Sørensen E.M. Lauridsen L. Margulies C. Mauricec Søren Schmidt, U.L. Olsen and D. Juul Jensena. Direct observation of 3-D grain growth in al-0.1% mn. *Scripta Materiala*, 59(5):491–494, 2008.

[6] M. Sambridge T. Bodin and K. Gallagher. A self-parametrizing partition model approach to tomographic inverse problems. *Inverse Problems*, 25(5):055009, 2009.

[7] Camilla H. Trinderup. Ray tracing problems for tomographic reconstruction in materials science. Master's thesis, Technical University of Denmark, 2011.