

UNIVERSITEIT UTRECHT

MASTER THESIS FOR SCIENTIFIC COMPUTING

---

# Image reconstruction methods for emission based tomography

---

*Author:*  
Jeroen de WAARD

*Supervisors:*  
Gerard SLEIJPEN and  
Paul ZEGELING

August 22, 2011

# 1 Preface

This thesis was written in a time span of 4 years and it would never had finished if not for Gerard's patience and his valuable comments. I also want to thank Casper, Sietske and Rianne for their time and invaluable support. Furthermore I would like to thank the hospital Rivierenland Tiel for showing me their SPECT scanner and their clinical physicist Leo Romijn for his explanation of the medical side of the subject.

## Contents

<b>1</b>	<b>Preface</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Image Reconstruction</b>	<b>3</b>
3.1	The structure of $M$ . . . . .	5
3.2	Singular Value Decomposition . . . . .	6
3.3	Statistics . . . . .	7
<b>4</b>	<b>Numerical Methods</b>	<b>9</b>
4.1	Filtered Backwards Projection . . . . .	9
4.2	MLEM . . . . .	11
4.2.1	Computational Cost . . . . .	12
4.3	OSEM . . . . .	12
4.4	Least Squares . . . . .	12
4.4.1	Computational Cost . . . . .	13
4.5	Least Squares with substitution . . . . .	14
4.5.1	Computational Cost . . . . .	14
4.6	Newton-Raphson . . . . .	15
4.6.1	Computational Cost . . . . .	16
4.7	Non linear CG . . . . .	16
4.7.1	Computational costs . . . . .	17
4.8	Example . . . . .	17
<b>5</b>	<b>Test results</b>	<b>20</b>
5.1	The simulation . . . . .	20
5.2	simple Preconditioning . . . . .	22
5.3	Stopping Conditions . . . . .	23
5.4	Effects of the number of projection points . . . . .	24
5.5	Performance of MLEM . . . . .	26
5.6	The relation between grid size and photon count . . . . .	27
5.7	Comparison between MLEM and LS . . . . .	27
5.8	Brute Force . . . . .	29
5.9	The effect of substitution on the convergence of CGLS . . . . .	29
5.10	performance of the Newton-Raphson method . . . . .	30

5.11 Effect of Radiation . . . . .	30
5.12 Convergence for different foci . . . . .	30
<b>6 Conclusion</b>	<b>31</b>
<b>A Matlab source code</b>	<b>31</b>
A.1 Main file . . . . .	31
A.2 Constructing the matrix M . . . . .	34
A.3 Constructing the tracer distribution . . . . .	39
A.4 The MLEM method . . . . .	40
A.5 The LS method . . . . .	41
A.6 The LS method with substitution . . . . .	42
A.7 The Newton Raphon method . . . . .	42

## 2 Introduction

This thesis deals with the construction of images based on data derived from medical scanners. The focus will lie at the reconstruction of 3D images from data generated by an U-spect scanner. U-Spect stands for Ultra-single Photon Emission Computed Tomography and is an improved version of a Spect scanner. Both are similar to PET-scanners, which stands for Positron Emission Tomography. Recently an even newer version of the U-Spect scanner was introduced, see [10] for a comparison between U-Spect I and II.

At the time of writing U-Spect is in development and not used for medical purposes. Like PET, the Spect scanner uses radioactive tracers which are injected in vitro and bind with specific molecules and thus cells. This feature is the main difference with more traditional types scans like CT, which give a 3D image based on several X-ray projections. Choosing the right tracer enables the localization of specific cells and as active cells bind more tracers, activity can be measured. This makes PET and Spect functional scans.

The core difference between PET and Spect is that a Spect scanner uses a different kind of radioactive isotope leading directly to gamma radiation, while the isotope used in PET scanners sends a positron. A positron is annihilated by an electron which generates two gamma rays in opposite directions. The image detection of Spect scanners is usually combined with a CT scan to get higher resolution images.

Traditional Spect scans have a poor resolution compared to PET or CT scans because only a tiny fraction of the radiation is detected. The main advantage of Spect is that it uses isotopes which are easier to work with and cost less than those for PET. Recent problems with nuclear facilities like the plant in Petten underline the need for easier to use isotopes. There is huge demand to increase the accuracy of Spect scanners and with this in mind the U-Spect was developed.

This thesis compares several algorithms to construct images. Though in principal all emission based scanners produce a similar equation, the actual parameters depend on the type of scanner. To test the algorithms a simulation is used to provide the projection data. Seeing the U-Spect scanner as a promising development, the set-up of the U-Spect scanner is used as a model for the simulation.

## 3 Image Reconstruction

Before a scan starts a tracer is injected into the patient or animal. The tracers consist of an radioactive isotope within a molecule that has a high probability to bind with specific cells. During the scan the isotopes decay and send out gamma radiation. This radiation is caught by a gamma detector. Which is basically a scintillating crystal which turns the radiation into a flash of light. These events are counted and turned into a two dimensional picture. Unlike PET scans that produce a pair of (gamma) photons, it is not possible to determine the direction

of a single photon. Without knowing where the detected photon originated from its detection contains no information. To deal with this problem Spect scanners have a collimator dotted with pinholes around the source of the radiation. As the photons can only reach the gamma detector through these pinholes, we know, within a small margin, their direction at detection.

Enforcing a direction on the measured photons can be achieved in several ways. One is to use a tube which only allow photons to pass perpendicular to the gamma detector. The other is to use pinholes through which all photons must pass. Traditional Spect scanners use tubes, U-Spect scanners use pinholes. The number of pinholes in the collimator determines the amount of two dimensional images which are simultaneously taken. Both methods have the drawback that most photons are blocked by the collimator. Using tubes has the drawback that only one tube can register the photons from the scanning region, whereas the viewing area of pinholes will overlap generating several measured values from the same area. Figure 1 shows a schematic cross section of the scanner.

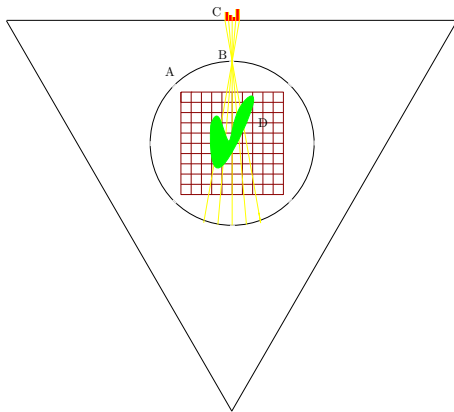


Figure 1: Cross-section of the system

The red grid in the middle, D, is surrounded by an impenetrable cylinder, A, the collimator with various pinholes like B in the picture. Radiation from the grid passes these pinholes and is projected onto a plate at C and grouped into several bins. Only radiation passing through the pinhole within a certain angular range can pass. In 3D the projection on the plate C forms a 2D picture and the values at the bins are represented by pixels. The concentration of tracers in the grid, shown as a green blob in figure 1 is then represented by 3D pixels, which are called voxels. Note that figure 1 is somewhat simplified, the concentration of tracers in this figure is either zero or one, represented by green. Normally the concentration of tracers and thus the colors used, have more variation.

Reconstruction of the image is hampered by an abundance of practical problems. We will ignore these and focus on a simplified simulation which we use to compare different algorithms for reconstructing the image. In this form the

problem can be described as follows.

The distribution of tracers or rather of isotopes depends on both space and time, and can be expressed by  $\bar{\lambda}(x)e^{-\beta t}$  with  $\beta$  the half-life of the isotope used. The integral over the duration  $T$  of the scan is  $\bar{\lambda}(x)(1 - e^{-\beta T})/\beta$ . It follows that there is linear cause between  $\bar{\lambda}_j$  and the concentration of tracers in each cell, which allows us to disregard time in our equation.

The portion of space in which the tracer can spread is discretized by dividing it into a grid of cells. We use the middle of each cell for the approximation of the entire cell. During the scan the tracers from each cell send out radiation. The amount of radiation from the  $j$ th cell during the entire scan is  $\lambda_j$  and together they form a voxel. A voxel is essentially as 3D pixel. The average amount of radiation during the scan is  $\bar{\lambda}_j$ . Note that the 3-dimensional array of cells is numbered by the one dimensional index  $j$ .

For each pinhole, the radiation is projected into a two dimensional plate. Each such projection is divided into an 2 dimensional array of bins.  $p_i$  the data within these bins and forms pixels, note that the 2-dimensional projections are numbered by the one dimensional index  $i$ .

The relation between which cells project to which bin is described by a matrix which we will call  $M$ . Each element  $M_{ij}$  gives the portion of radiation in cell  $j$  which will reach bin  $i$ . If we call  $\lambda$  a discretization of the unknown distribution of tracers and  $p$  the measured projection data then the problem can be seen as solving

$$M\lambda = p. \tag{1}$$

We choose the number of cells, and thus unknowns, such that it is smaller than the number of bins, and thus projection data. This makes  $M$  non-square. Because each bin sees only a small portion of the number of cells the matrix  $M$  is also sparse. Choosing a square grid to discretize the unknown distribution of  $\lambda$  is not ideal if we want to keep the number of non zeros at a minimum and the unknowns decoupled from the others. But it was chosen for programming simplicity.

### 3.1 The structure of $M$

Calculations to approximate the distribution of tracers will only use the non zero elements of  $M$ . To get an estimate for the number of non zero matrix elements, we call  $d$  the resolution and define it as the number of grid elements used in one dimension. In  $\mathfrak{R}^2$  the number of matrix elements is in the order of  $d^4$  but the amount non zero elements is  $d^3$  as on average only in the order of  $d$  grid elements are intersected by the rays from each projection bin, the yellow line in figure 1. Likewise the number of non zero elements in  $\mathfrak{R}^3$  is in the order of  $d^4$  while the total number of elements is in the order of  $d^6$ . We conclude the matrix becomes very sparse in three dimensions in the sense that the number of non zero elements is much smaller than the total amount of elements.

The complexity of all iterative methods depends on the number of non-zero elements which rise quickly as we require a higher resolution. If we narrow the

visible range of each collimator by a certain fraction, the projected images are the result from smaller portions of the grid and the matrix  $M$  becomes more decoupled. As a result the amount of non-zero elements is reduced. Unfortunately the projection data suffers from a relatively larger error since it is derived from less photons.

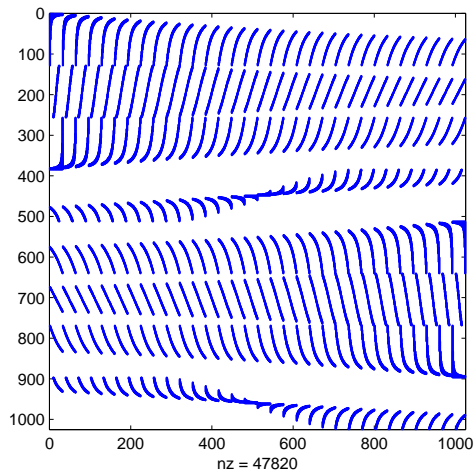


Figure 2: Matrix M

Figure 2 shows the non zero elements of the matrix for a system in two dimensions with a resolution  $d$  of 32 grid elements for a total of 1024 grid elements. The number of pinholes is 8 and each projects an image into 128 bins. The resulting matrix  $M$  is square and has 1024 by 1024 elements with just under 50 thousand non zero. The number of elements of  $M$  is  $d^4$  and the number non-zero elements is in the order of  $d^3$  as we claimed earlier.

This configuration results in a matrix  $M$  with two degrees of symmetry. This symmetry is caused by placing the pinholes equiangular around the grid. The rays from one pinhole with the grid, like the yellow lines in figure 1, overlap the grid elements with the same ratio as the rays from a pinhole  $\pi/2$  radian further. This property can be used to store only a quarter of the elements and thereby reducing the memory costs. Doing so however would make the code more complex and the placement of pinholes less flexible. As memory costs were not an issue for the relative small tests performed for this thesis, the symmetry of  $M$  was not used.

### 3.2 Singular Value Decomposition

A singular value decomposition (SVD) of  $M$  can be used to analyze the singularity of the matrix. The decomposition factorizes the matrix as the product of three matrices  $U\Sigma V^*$ .  $U$  and  $V$  form orthonormal bases and  $\Sigma$  is a diagonal

matrix with the singular values on its diagonal. The SVD of  $M$  is related to the eigendecomposition of  $MM^*$  as the singular values of  $M$  are the square roots of eigenvalues of  $MM^*$ . Therefore the ratio of the largest singular value and the smallest tells us how badly conditioned the matrix  $MM^*$  is.

For the square matrix  $M$  in figure 2 none of the singular values is equal to zero but the smallest is  $7.9883e-022$  and with a machine precision of  $2.2204e-16$  this makes the matrix singular. If we double the amount of pixels for each pinhole to 256 the largest singular value is  $3.0700e-005$  while the smallest is  $2.1965e-008$ , which is well above the machine precision.

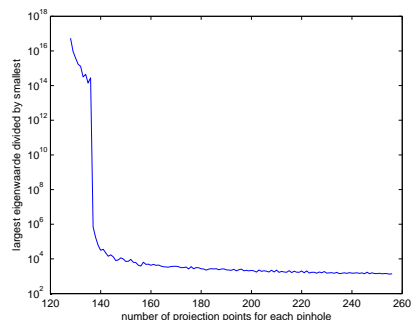


Figure 3: The condition of  $MM^*$

Figure 3 shows the largest singular value divided by the smallest for matrix  $M$  with 32 by 32 grid elements and 8 times 128 to 256 projections points. Though we would like the number of grid points as large as possible, as this minimizes the discretization error, the figure shows that choosing the number of grid points equal to the number of projection points will lead to a badly conditioned matrix. There is a sharp decline at 138 and one might see this as an optimal number for a grid of 32 by 32 and the chosen configuration of pinholes.

Later in paragraph 5.4 we will look into the effects of increasing the number of projection points on the quality of the resulting approximation.

### 3.3 Statistics

The observed data  $p$  is a projection of the scanning region into several planes. Determining the unknown distribution of tracers  $\lambda$  is not simply a matter of solving  $M\lambda = p$ .

Both  $\lambda$  and  $p$  lie in a separate sample space, say  $\Lambda$  and  $P$ . Mathematically, performing a scan is equal to a mapping of  $\lambda$  to  $p(\lambda)$  from  $\Lambda$  to  $P$ . Due to the stochastic nature of radiation a large range of  $\lambda$  can be mapped into the observed  $p$ . Thus the mapping from  $\Lambda$  to  $P$  is many to one. Due to this characteristics we call the observed  $p$  incomplete data and  $\lambda$  the complete data. Dempster et al. [3] developed a general algorithm for problems with incomplete



data. This algorithm is called the Expectation Maximization (EM) algorithm. Its first step is to compute the expectation and the second step is to compute the set of parameters that maximizes the expectation.

Our emission based problem is a specific incomplete data problem. Shepp et al. developed in [9] an algorithm to deal with emission based tomography. Before we can introduce this algorithm we need to explain more about the statistical nature of emission based tomography.

Radiation is a classical example of a Poisson distribution. The probability mass function of the Poisson distribution is given by

$$f(a, b) = \frac{b^a e^{-b}}{a!}, \quad (2)$$

as a function of the integer  $a$  with mean  $b$ . The sum of two Poisson distributed variables with mean  $\lambda_1$  and  $\lambda_2$  is also a Poisson distribution and has mean  $\lambda_1 + \lambda_2$ .

A gamma particle resulting from the decay of the isotope has no preference for its direction, so each particle has a small and independent chance to be detected within a certain set of directions. We call  $X$  the amount of gamma particles originating from a volume in the grid and  $Y$  the amount detected in a certain region of the gamma detector.

The variable  $X$  has a Poisson distribution, we call its mean  $\lambda$ . The random variable  $Y$  is the sum of  $X$  Bernoulli variables. We call  $p$  the chance to contribute and  $q = 1 - p$  the chance a particle does not contribute.

For a simulation of the scan we need to know the compound distribution of  $Y$ .

The chance for random variable  $Y$  to be equal to an integer  $j$  is

$$\begin{aligned} \sum_{n=0}^{\infty} P(X = n)P(Y = j|X = n) &= \sum_{n=0}^{\infty} \frac{e^{-\lambda} \lambda^n}{n!} \binom{n}{j} p^j q^{n-j} \\ &= \frac{e^{-\lambda} \lambda^j p^j}{j!} \sum_{n=j}^{\infty} \frac{(\lambda q)^{n-j}}{(n-j)!} = \frac{e^{-\lambda} (\lambda p)^j}{j!} (1 + \lambda q + \frac{(\lambda q)^2}{2!} + \dots) \\ &= \frac{e^{-\lambda} (\lambda p)^j}{j!} e^{\lambda q} = \frac{e^{-\lambda p} (\lambda p)^j}{j!} \end{aligned}$$

This last expression is the chance for a Poisson distributed variable with mean  $\lambda p$  to be equal to  $j$ . Thus the amount of registered photons is Poisson distributed with mean  $\lambda p$ , which is intuitive.

We can now state that the measured value  $p_j$  is a Poisson variable with mean  $\bar{p}_j$  and its mean is given by:

$$\bar{p}_i = \sum_j M_{ij} \bar{\lambda}_j. \quad (3)$$

The chance of measuring  $p_j$ , with mean  $\bar{p}_j$ , is given by the Poisson distribution,

$$P(p_i) = \frac{\bar{p}_i^{p_i} e^{-\bar{p}_i}}{p_i!}. \quad (4)$$

The likelihood function  $L(\bar{\lambda}|\mathbf{p})$  is equal to  $P(\mathbf{p}|\bar{\lambda})$ . Because all detected values in each bin are independent we can write the likelihood function as,

$$L(\bar{\lambda}|\mathbf{p}) = P(\mathbf{p}|\bar{\lambda}) = \prod_i P(p_i) = \prod_i \frac{\bar{p}_i^{p_i} e^{-\bar{p}_i}}{p_i!}. \quad (5)$$

We are interested in the  $\bar{\lambda}$  which maximizes the likelihood. For this we differentiate  $l(\bar{\lambda}) \equiv \ln(L(\bar{\lambda}))$ .

$$l(\bar{\lambda}) = \sum_i (-\sum_j M_{ij} \bar{\lambda}_j + p_i \ln(\sum_j M_{ij} \bar{\lambda}_j) - \ln(p_i!)) \quad (6)$$

and set the derivative to zero,

$$\frac{\partial l(\bar{\lambda})}{\partial \bar{\lambda}_j} = -\sum_i M_{ij} + \sum_i M_{ij} \frac{p_i}{\sum_j M_{ij} \bar{\lambda}_j} = 0. \quad (7)$$

We want to find those  $\bar{\lambda}_j$  for which the above equations hold true. By introducing a new notation these can be rewritten as

$$M^T \text{diag}^{-1}(M\bar{\lambda})p = M^T \mathbf{1}. \quad (8)$$

With  $\mathbf{1}$  a vector of ones and  $\text{diag}(x)$  a diagonal matrix with  $x_i$  as the  $i$ -th diagonal element.

We can rewrite this equation to

$$M^T \text{diag}^{-1}(M\bar{\lambda})M\bar{\lambda} = M^T \text{diag}^{-1}(M\bar{\lambda})p, \quad (9)$$

which we will later use to derive a numerical algorithm. Note that in case  $M$  is square and  $M^{-1}$  exists, equation (8) is equivalent to  $M\bar{\lambda} = p$ .

## 4 Numerical Methods

### 4.1 Filtered Backwards Projection

Historically, tomography problems were solved by a method called filtered back projection. Filtered back projection consists of an inverse Radon transformation on the projections with a high-pass frequency filter on the constructed image to compensate for blurring. A Radon transformation in two dimensions is simply the integral of a function  $f: \mathfrak{R}^2 \rightarrow \mathfrak{R}$  over a line  $L$ .

A lot of medical scans use sensor around encircling a finite area of interest, which detects the sent radiation. With this set-up it makes sense to parametrize the line  $L$  and use  $L_{s,\alpha}(t) = ((t \sin \alpha + s \cos \alpha), (-t \cos \alpha + s \sin \alpha))$  with  $\alpha$  the angle between the projection line and the positive x-axis and  $s$  its distance to

the origin. Figure 4 shows this set-up, the green area is the distribution of particles which sent out radiation, the red bars are the line integrals over the distribution.

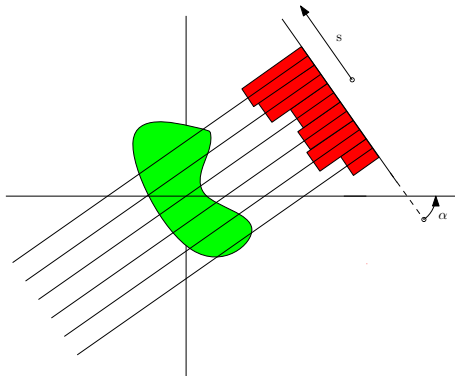


Figure 4: A typical set-up for a Radon transformation

With these parameters the two dimensional Radon transform becomes,

$$Rf(\alpha, s) = \int_{-\infty}^{\infty} f((t \sin \alpha + s \cos \alpha), (-t \cos \alpha + s \sin \alpha)) dt.$$

We assume the measured projection to be the line integral of the unknown distribution. The projections for different angles and distances  $s$  can be seen as a function  $g$  from  $\mathfrak{R}^2 \rightarrow \mathfrak{R}$  with  $\alpha$  and  $s$  as variables, this data is called a sinogram because of its typical sinus form.

The dual of the 2D Radon operator also called backprojection operator of a function  $g$  is,

$$(R^*g)(x) = \frac{1}{2\pi} \int_{\alpha=0}^{\alpha=2\pi} g(\alpha, x_1 \cos \alpha + x_2 \sin \alpha) d\alpha.$$

Using this formula the unknown distribution can be calculated from the measured projections. What basically happens is that the measured values are distributed evenly over that part of the projection lines which lie in the area of interest. This process produces a lot of blurring so a filter must be used to dampens the low frequencies. The backprojection operator is therefore not used directly on the projections  $Rf(\alpha, s)$  but on the convolution of  $Rf(\alpha, s)$  with an appropriate filter  $g$ . The tomography entry in [8] has a more detailed explanation.

Filtered backward projection was used to reconstruct emission based tomography until more suitable methods were derived which take into account the errors produced by the statistical nature of the radiation. The set-up studied in this thesis deals with scans using a camera obscura with fixed cameras and the set-up described with filtered backwards projection uses moving cameras, or rather projections, with a narrow viewing angle.

Filtered back projection is still widely used for transmission based scans involving X-rays where the static nature of radiation does not play a significant role, see [1].

Though it would be possible to derive formulas for filtered backward projection using fixed cameras with a wider viewing angle, the filtered back projection was not implemented.

The previous chapter shows that the solution of equation (8) is the most likely. The non-linearity prevents us from using a fast converging algorithm like CG. The next paragraphs will deal with several algorithms to solve equation (8). To compare the outcome of these algorithms we look at the norm of the difference between the approximation from the algorithms and the exact solution. We prefer to compare the outcome with the minimum variance unbiased estimator which is simply the sample mean of the radiation from each grid element. Just as the exact solution, the sum of radiation from each grid element is not known in practice as these variables are added with radiation from other elements and this sum is detected by the gamma detector.

## 4.2 MLEM

In Section 3 we saw that the statistical error in  $p$  changes the problem from  $M\lambda = p$  into equation (8). To construct an algorithm to solve this equation, we can multiply equation (7) by  $\bar{\lambda}_j$ , in this form iteration indices can be introduced to get

$$\lambda_j^{k+1} = \frac{\lambda_j^k}{\sum_i M_{ij}} \sum_i \frac{M_{ij} p_i}{\sum_j M_{ij} \lambda_j^k}. \quad (10)$$

We refer to this iteration as the Maximum-Likelihood Expectation-Maximization Method or MLEM method in short. In matrix-vector notation this iteration becomes,

$$\lambda^{k+1} = \text{Diag}(M^T \mathbf{1})^{-1} \text{Diag}(p^T \text{Diag}(M\lambda^k)^{-1} M) \lambda^k. \quad (11)$$

Equation 10 can be seen as applying a projection of the old  $\lambda$  and the back projection of this projection to get a new  $\lambda$ . The projection step is,

$$\mu_i^k = \sum_j M_{ij} \lambda_j^{(k)}.$$

The back projection is,

$$\lambda_j^{k+1} = \frac{\lambda_j^{(k)}}{\sum_i M_{ij}} \sum_i \frac{M_{ij} p_i}{\mu_i^k}.$$

Csiszar et give a technical proof of convergence in their paper [2] which can be applied to (8). The author of [7] claims that the MLEM-method will converge similar to a SIRT method.

### 4.2.1 Computational Cost

The pseudo code for the MLEM method is shown in algorithm 1.

```

 $\lambda^0 = \mathbf{1}$ 
for  $k = 1$  to maxiter do
     $\lambda^{k+1} = \lambda^k \cdot (M' \cdot (p \cdot ((M \cdot \lambda^k)^{-1})))$ 
end for

```

**Algorithm 1:** Pseudo code for MLEM

The computational costs for each iteration are two matrix-vector multiplications and 4 vector-vector multiplications. The matrix  $M$  is sparse and the average number of non-zeros in 2D is in the order of  $\sqrt{n}$  with  $n$  the number of cells of the grid. Thus the computational costs are in the order of  $n\sqrt{n}$  per iteration. In practice the speed of this algorithm will largely depend on the speed of convergence as the costs per iteration step are small.

### 4.3 OSEM

To increase the convergence speed of the MLEM algorithm a modified version was developed called OSEM. OSEM, Ordered Subsets Expectation Maximization, orders the projection data into groups and subsequently iterates once with the standard MLEM-algorithm on each set before starting over. One compare the way OSEM differs from MLEM as SIRT methods differ from ART. The groups can overlap and it is suggested in [6] to take (nearly) orthogonal projections as sets. The same paper also shows that OSEM converges an order faster then standard EM. OSEM was not implemented as it is just a faster version of MLEM.

### 4.4 Least Squares

There is no  $\lambda$  such that  $M\lambda = p$ , in other words the vector  $p$  is not in the column space of  $M$ . This is easily understood as the matrix  $M$  has fewer columns then rows and on top of that the vector  $p$  is perturbed with a Poisson distributed random vector. So instead of trying to find a solution of the equation  $M\lambda = p$ , one can try to find a  $\lambda^*$  such that it minimizes  $\|p - M\lambda\|_2$ . What we would like to find is the point  $\lambda_{LS}$  in  $C(M)$  which is closest to  $p$ . This  $\lambda_{LS}$  is the projection of  $p$  onto the column space of  $M$ . Thus

$$M\lambda^* = \text{proj}_{C(M)}(p).$$

As  $\text{proj}_{C(M)}p - p$  is perpendicular to  $C(M)$  it follows that  $M\lambda^* - p$  is perpendicular to  $C(M)$  and thus multiplying with  $M^T$  yields,

$$M^T(M\lambda^* - p) = 0.$$

This leads to the so called the normal equations,

$$M^T M \lambda^* = M^T p. \tag{12}$$

A slightly modified version CG can be used to get a solution of equation (12). Solving equation (12) does not solve  $M\lambda = p$  in a strict sense but rather gives a solution which minimizes the norm of the residue. The least square method has some drawbacks, particularly in our case.

$M$  is a sparse matrix and multiplying with  $M^T$  results in a lot of fill-in. This fill-in can be avoided by not calculating  $M^T M$  explicitly but rather use two matrix vector multiplications instead of one in the CG algorithm. Another inconvenience is that CG converges at a rate of the square root of the condition number of the matrix  $= \sqrt{\kappa}$ . The condition number of  $M^T M$  is that of  $M$  squared thus CG converges at a rate of  $\kappa$ .

These two drawbacks only compromise the speed of convergence. The real drawback, which needs to be addressed, is that the least squares method does not correctly take into account the statistical nature of the errors. The errors in the measured data do not follow the normal distribution and thus the least squares method does not follow the maximum likelihood criterion. Furthermore the relatively low amount of samples, in the order of the amount of pinholes, can lead to solutions outside the feasible range i.e. negative. The method of maximum likelihood estimation is tailored to the Poisson nature of the variables and one might expect better results with it.

In case  $M$  is square and invertible, the solution of (12) is the same as the solution of  $M\lambda = p$ . As we have seen at the end of chapter 3, this solution is also the same as the solution from (8).

#### 4.4.1 Computational Cost

CG can be used to solve equation (12). This slightly adjusted CG method is called CGLS, CG Least Squares.

```

 $\lambda_{LS} \leftarrow CGLS(M, \lambda_0, r_0)$ 


---


 $\rho_0 = r_0$ 
for while  $r > threshold$  do
   $\kappa_k = M\rho_k$ 
   $\alpha = \frac{r_k^T r_k}{\kappa_k^T \kappa_k}$ 
   $\lambda_{k+1} = \lambda_k + \alpha_k \rho_k$ 
   $r_{k+1} = r_k - \alpha_k M^T \kappa_k$ 
   $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ 
   $\rho_{k+1} = r_{k+1} + \beta_k \rho_k$ 
end for

```

**Algorithm 2:** Pseudo code for CGLS

The computational costs are 2 sparse matrix- vector multiplications each iteration. This is about twice as much as standard CG.

## 4.5 Least Squares with substitution

The method of least squares does not solve equation (8) but rather attempts to minimize the residue of  $M\lambda = p$ .

Still least squares can be used as a first step in cases where the error introduced by the Poisson variables is relatively small.

Introducing the notation  $M_\lambda = \text{diag}(M\lambda)^{-\frac{1}{2}}M$  and  $p_\lambda = \text{diag}(M\lambda)^{-\frac{1}{2}}p$  in equation (9). This equation becomes

$$M_\lambda^T M_\lambda \lambda = M_\lambda^T p_\lambda. \quad (13)$$

Introducing indices we can write

$$M_{\lambda_k}^T M_{\lambda_k} \lambda_{k+1} = M_{\lambda_k}^T p_{\lambda_k}. \quad (14)$$

For  $k = 0$  we take  $\text{diag}(M\lambda_0)^{-\frac{1}{2}} = I$  and equation (14) becomes identical to solving  $M\lambda = p$  with least squares. This will be the first iteration. Note that we do not need to know  $\lambda_0$  explicitly.

Equation (13) can be seen as the least squared system multiplied with  $D = \text{diag}(M\lambda)$ . This introduces a non-linear term in the equation which makes this system more difficult to solve.

### 4.5.1 Computational Cost

The pseudo code for least squares with substitution is shown in algorithm 3.

```

M0 = M
p0 = MTp
λ0 = 0
for k = 0 to maxiter do
  rk = MTp - MTMλk
  λk+1 = CGLS(Mk, λk, rk)
  Dk+1 = diag(Mλk+1)-1/2
  Mk+1 = Dk+1M
  pk+1 = Dk+1pk
end for

```

**Algorithm 3:** Pseudo code for LS with substitution

The computational costs are  $k$  times the costs of Least squares. In the implementation of the algorithm we keep track of the matrix  $\text{diag}(M\lambda_{k+1})^{-\frac{1}{2}}M$  resulting in roughly twice as much memory costs as with normal CGLS. Alternatively we can construct the elements of  $M_k$  when needed thus reducing the memory costs but increasing the complexity of the implementation and its computation costs. For each call to the least squares subroutine we assume it stops when it reaches its best approximation.

## 4.6 Newton-Raphson

One of the best known methods to find the solution of a non-linear equation like (9) is Newton-Raphson. In one dimension the method is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad (15)$$

$f(x)$  is the function for which we want to find a root and  $x_n$  a sequence of approximations, which hopefully converges to a solution. On the condition that the starting point is close enough to a solution and the derivative is not zero, the method converges quadratically. The method of Newton-Raphson can be generalized for  $n$  dimensions and becomes

$$\lambda_{t+1} = \lambda_t - J^{-1}(\lambda_t)F(\lambda_t). \quad (16)$$

For simplicity we use equation (8) for the function  $F(\lambda)$ . The Jacobian in  $\lambda$  is the linear term in  $\epsilon$  of  $F(\lambda + \epsilon) - F(\lambda)$ . With

$$F(\lambda) = -M^T \text{diag}(M\lambda)^{-1} \mathbf{p} + M^T \mathbf{1}. \quad (17)$$

Now we can write

$$\begin{aligned} F(\lambda + \epsilon) - F(\lambda) &= M^T [\text{diag}(M\lambda)^{-1} - \text{diag}(M\lambda + M\epsilon)^{-1}] \mathbf{p} = \\ &= M^T [\text{diag}(M\lambda)^{-1} \text{diag}(M\lambda + M\epsilon)^{-1} [\text{diag}(M\lambda + M\epsilon) - \text{diag}(M\lambda)]] \mathbf{p} = \\ &= M^T [\text{diag}(M\lambda)^{-1} [\mathbf{I} - \text{diag}(M\lambda) \text{diag}(M\lambda + M\epsilon)^{-1}]] \mathbf{p} = \\ &= M^T [\text{diag}(M\lambda)^{-1} [\mathbf{I} - (\mathbf{I} + \text{diag}(M\epsilon) \text{diag}(M\lambda)^{-1})^{-1}]] \mathbf{p}. \end{aligned}$$

Using  $(\mathbf{I} + E)^{-1} = \mathbf{I} - E + E^2 + (\dots)$  with  $E = \text{diag}(M\epsilon) \text{diag}(M\lambda)^{-1}$  and ignoring the terms with quadratic or higher powers of  $\epsilon$ , we get,

$$J(\lambda)\epsilon = M^T \text{diag}(M\lambda)^{-2} \text{diag}(M\epsilon) \mathbf{p}.$$

Using  $\text{diag}(M\epsilon) \mathbf{p} = \text{diag}(\mathbf{p}) M \epsilon$  the Jacobian becomes,

$$J(\lambda) = M^T \text{diag}(M\lambda)^{-2} \text{diag}(\mathbf{p}) M. \quad (18)$$

This formula shows that the Jacobian is symmetric. Elements wise the Jacobian is,

$$J_{ij} = \sum_{k=1}^m \frac{M_{ki} p_k M_{kj}}{(\sum_{s=1}^n M_{ks} \lambda_{t,s})^2}. \quad (19)$$

We can rewrite (16) as

$$J(\lambda_t) * \Delta \lambda_{t+1} = -F(\lambda_t), \quad (20)$$



with  $\Delta(\lambda_{t+1}) = \lambda_{t+1} - \lambda_t$  and use CGLS to avoid computing  $J^{-1}$ . Just as with CGLS the elements from the non sparse Jacobian can be derived when needed instead of stored, eliminating the need to store the matrix, at a computational cost.

#### 4.6.1 Computational Cost

Using the method of Newton-Raphson we solve (20) to get a sequence  $\lambda_t$  which converges to a solution  $\lambda_{NR}$ . Solving the equation is done with CGLS in a way very similar to CGLS with substitution. To avoid computing the Jacobian explicitly we write  $J_\lambda = M_\lambda^T M_\lambda$  with  $M_\lambda = \text{diag}(M\lambda)^{-1} \text{diag}(p)^{1/2} M$ . Likewise the right hand side of equation  $J(\lambda_t)\Delta_t = -F(\lambda_t)$  is substituted to  $p_\lambda = p^{1/2} - \text{diag}(M\lambda) * p^{-1/2}$ , with a dot denoting an element wise operation. Using  $p_\lambda$  and  $M_\lambda$  in the least square notation yields  $M_\lambda^T M_\lambda \Delta\lambda_{t+1} = M_\lambda^T p_\lambda$  which is equivalent to  $J(\lambda_t)\Delta_t = -F(\lambda_t)$ . Because the Jacobian is not defined for  $\lambda = 0$  the first iteration will be with  $M_\lambda = M$  and  $F(\lambda) = M^T p$ . This makes the first Newton-Raphson step equal to least squares.

```

 $\lambda_{NR} \leftarrow NR(M, p, \lambda_0)$ 


---


 $\lambda_0 = 0$ 
 $M_{\lambda_0} = M$ 
 $b_0 = p$ 
for  $t = 0$  to  $t_{max}$  do
   $\Delta = CGLS(M_{\lambda_t}, \lambda_t, b_t)$ 
   $\lambda_{t+1} = \lambda_t + \Delta$ 
   $M_{\lambda_{t+1}} = \text{diag}(M\lambda_{t+1})^{-1} \text{diag}(p)^{1/2} M$ 
   $b_{t+1} = p^{1/2} - \text{diag}(M\lambda_{t+1}) * p^{-1/2}$ 
end for

```

**Algorithm 4:** Pseudo code for Newton Raphson

## 4.7 Non linear CG

The method of CG can easily be adjusted to solve non-linear problems. The outline of non linear CG is to start by setting the initial search direction  $d_0$  towards the steepest decent  $-\nabla f(\lambda_0)$ , where  $f(\lambda)$  is equation (8).

The approximation  $\lambda_i$  is then updated in the search direction  $d_i$  by the optimal factor  $\alpha$ . Unlike the linear case  $\alpha$  can not be simply computed. A line search which minimizes  $f(\lambda_i + \alpha d_i)$  is used to get  $\alpha$ .

The new search direction is set to minus steepest plus  $\beta$  times the old search direction. With linear CG,  $\beta$  is set to  $\frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$ , this is called the Fletcher-Reeves formula. The Fletcher-Reeves formula can be used with non linear CG, but the so called Polak-Ribière formula is more popular due to faster convergence:

$$\beta^{PR} = \frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i}. \quad (21)$$

In case  $\beta^{PR} < 0$  Polak-Ribière might not converge. When this happens the CG sequence is restarted by taking  $\beta = 0$  and thus setting the search direction to the residue. Note that in case the system of equations is linear  $r_{i+1}^T r_i = 0$  as CG makes each the residue direction orthogonal to its previous directions. With linear systems the Polak-Ribiere parameter is equal to Fletcher-Reeves.

The paper [5] gives an overview of the most commonly used formulas for  $\beta$ . Note that non linear CG deals with an unconstrained problem whereas our problem is constrained because of the constraint  $\lambda_i \geq 0, \forall i$ . Due to time constraints and the poor performance of CGLS with substitution and Newton-Raphson, implementation of non-linear CG was not done.

#### 4.7.1 Computational costs

```

λ0 = 1
r0 = ∇(M, λ0, p)
d0 = -r0
for i = 0 to imax do
  ri+1 = ∇(M, λi, p)
  βt+1 =  $\frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$ 
  di+1 = -ri+1 + βi+1di
  αi ← Linesearch(M, p, λi, d)
  λi+1 = λi + αidi
end for

```

**Algorithm 5:** Pseudo code for Non Linear CG

## 4.8 Example

In general the solution of least squares equation (12) and the MLEM equation (8) is different for non square  $M$ . To illustrate this take

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

This  $M$  maps the relation of two unknown variables  $\lambda$  and three observed values  $p$ .  $M_{ij}$  is defined as the chance a hit is detected in the  $i$ -th pixel from the  $j$ -th cell. Normally  $M_{ij}$  would be small and to observe a decent amount of hits,  $\lambda$  is big. In this example  $M$  is multiplied and  $\lambda$  divided by a large factor. This is solely done to get more convenient numbers for  $M_{ij}$  and  $\lambda_i$ .

For this  $M$  least squares becomes

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} p_1 + p_3 \\ p_2 + p_3 \end{pmatrix},$$

with solution  $\lambda_{LS} = \begin{pmatrix} \frac{2p_1 - p_2 + p_3}{3} \\ \frac{2p_2 - p_1 + p_3}{3} \end{pmatrix}$ . It is possible, though unlikely, that one of the  $\lambda_i$  becomes negative. When this happens we know the solution is not optimal as the solution should be positive. Replacing the negative component with 0 yields a more accurate solution but it will increase the residue of the least squares equation.

The MLEM form using M in equation (8) is

$$\begin{pmatrix} \frac{p_1}{\lambda_1} + \frac{p_3}{\lambda_1 + \lambda_2} \\ \frac{p_2}{\lambda_2} + \frac{p_3}{\lambda_1 + \lambda_2} \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

With solution  $\lambda_{MLEM} = \begin{pmatrix} \frac{p_1}{2} + \frac{p_1 p_3}{2(p_1 + p_2)} \\ \frac{p_2}{2} + \frac{p_2 p_3}{2(p_1 + p_2)} \end{pmatrix}$ . During calculation the condition  $p_i \neq 0$  is introduced. On this condition the solution is positive.

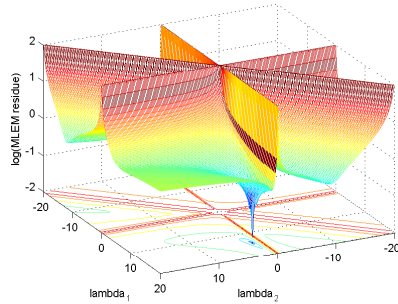


Figure 5: Residue of the MLEM representation as a function of  $\lambda$

For our sample  $M$  and with the very eccentric data  $p^T = (10, 1, 20)$ , figure 5 shows  $\|M^T \mathbf{1} - M^T \text{diag}^{-1}(M\lambda)\mathbf{p}\|_2$  on a logarithmic scale for all values in the square  $[-20, -20]$  and  $[20, 20]$  with a step size of 0.2. The norm is infinite on the lines  $\lambda_1 = 0$ ,  $\lambda_2 = 0$  and  $\lambda_1 = -\lambda_2$ , for displaying ease values above 2 were replaced by 2. With this  $p$  the exact solution is  $[155/11, 155/110]$ . The exact solution of the least square problem for this  $M$  and  $p$  is  $[13, 14/3]$ , which is very different from the the solution of the MLEM representation. The convergence plot for these parameters using MLEM method is shown in figure (6). This picture shows that even for this simple example, convergence is very slow, but it does convergences to the solution of equation (8).

The probability of observing  $p$  as a function of  $\lambda$ , given by equation (5), is displayed in figure 7, with  $\lambda \geq 0$ . The maximum is observed at  $[14, 1.4]$ . This shows that the MLEM representation finds the  $\lambda$  with maximum probability given  $p$  for this particular  $M$ .

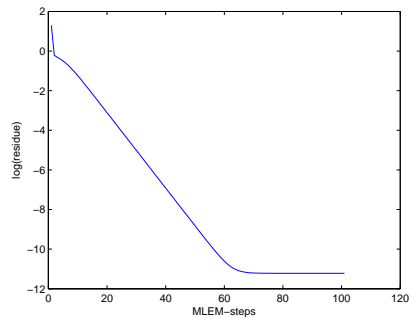


Figure 6: Residue of the MLEM method for each step

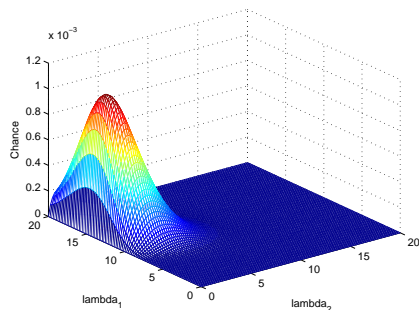


Figure 7: Chance as a function of  $\lambda$

If we take

$$M = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 1 \\ 1 & 1 \end{pmatrix}.$$

The same  $p$  as with the previous  $M$  would yield figure 8. Here the nearest solution of the MLEM representation is  $[15.6, -1.6]$ , which is not a feasible solution.

In conclusion we notice several conditions need to be met to ensure the MLEM representation gives a feasible solution.

These conditions are that no element of the vector  $M\lambda$  is 0 and that the observed data is not too eccentric or it will push the solution into an infeasible region.

The first can lead to disallowing the best solution. To counter this a shift of variables can be used. We can take  $\tilde{\lambda} = \lambda + d$ , with  $d$  equal to the entire amount of radiation. This also leads to an adjusted  $\tilde{p} = p + Md$ . To ensure the solution is at least equal to  $d$  a penalty function can be introduced raising the residue when the solution is smaller than  $d$ . The problem with the latter is that

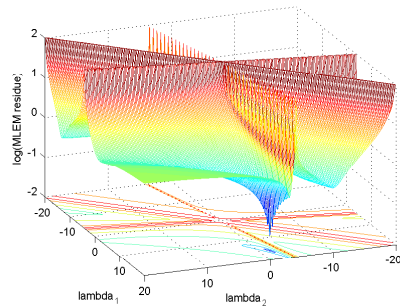


Figure 8: Residue of the MLEM representation as a function of  $\lambda$

sometimes an infeasible solution is the only solution.

## 5 Test results

### 5.1 The simulation

To test the various algorithms a simulation of the problem is used to generate the needed data. This simulation is a 2D version of the problem, as in figure 1. Because we want to focus on the mathematical problem of solving the unknown vector  $\lambda$  a number of simplifications were made:

- We assume all photons which collide with the gamma detector are detected.
- The projection plates are represented by a circle with the center of the scanner as center.
- Grid elements are squares and the grid itself is a square.
- Pinholes are regarded as points.
- Any quantum effects are disregarded.
- The projection plates are considered to be thin enough such that its thickness plays no role in determining where radiation hits the plate.
- Loss of radiation by absorption or scattering of the particles is not taken into account.
- The distribution of tracers is fixed during the scan, though their total volume can shrink.

In practice the matrix  $M$  is determined by placing a known amount of radioactive sample at various places in the scan area. This determines a matrix  $M$

in which most complications are incorporated. Absorption of radiation by the scanned body can be dealt with likewise. Essentially scattering and absorption leads to an adjustment of the values in the matrix  $M$ . Furthermore a gamma detector suffers from a cool down after a photon is detected before it can detect another photon. This leads to a non-linear relation between the amount of detected and actual radiation. Here we only focus on the method which gives the best approximations and disregard these complex refinements.

The variables which are taken into account in the simulation are:

- The positions of the pinholes
- Pinholes have a radius to determine the amount of radiation which passes through them
- The size of the grid and the radius of the collimator.
- The amount of radiation for one unit of tracers during the test
- The distribution of the tracer across the grid

For some tests the arbitrary smooth distribution

$$F(x, y) = C[\cos(2\pi x)\cos(2\pi y) + 1] \quad (22)$$

was chosen. Figure 9 shows the expected amount of radiation, per squared unit during the simulation. The parameter  $C$  is a large number to scale the function and represents the total radioactivity during the scan.

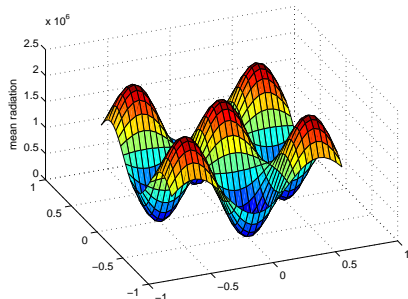


Figure 9: Distribution of the tracers

To measure how well the algorithms performs the error function

$$\epsilon = \|\bar{F} - \bar{f}\| \quad (23)$$

was used, with  $\bar{F}$  the vector with the values  $F(x, y)$  at the middle of the grid elements and  $\bar{f}$  the vector for the approximation function  $f$ .

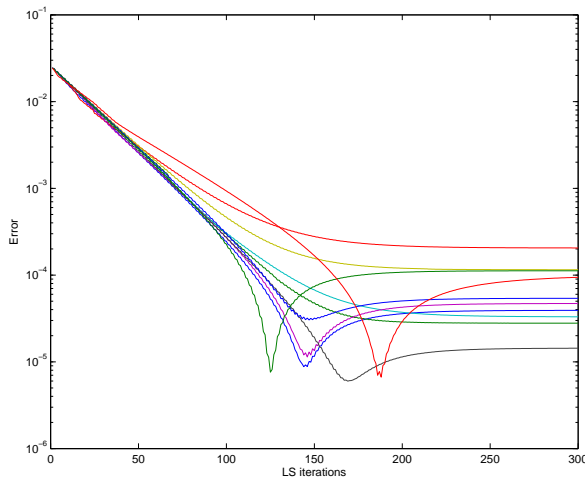


Figure 10: Convergence for different projections

## 5.2 simple Preconditioning

We can simplify the MLEM algorithm by disregarding the photons which are not detected. We call the associated matrix and unknown vector  $\tilde{M}$  and  $\tilde{\lambda}$  respectively. As all photons are detected it follows that  $\sum_i p_i = \sum_j \tilde{\lambda}_j$  and the column sum of  $\tilde{M}$  equals 1. If we define  $D_{jj} = \sum_i M_{ij}$ , the relations of  $M$  and  $\lambda$  are  $\tilde{M} = MD$  and  $\tilde{\lambda} = D^{-1}\lambda$ . This new system behaves the same as  $M\lambda = p$  in the sense that  $\tilde{p}_i = \sum_j \tilde{M}_{ij}\tilde{\lambda}_j$ . The adjusted MLEM sequence is equal to the original system. The difference is that the found approximation  $\tilde{\lambda}$  differs from  $\lambda$  by a factor  $D$ , which we need to use to scale the approximation after calculation.

The original MLEM algorithm derived by Shepp et al. in [9] makes use of this simplification, which leads to

$$\tilde{\lambda}_j^{k+1} = \tilde{\lambda}_j^k \sum_i \frac{M_{ij}p_i}{\sum_j M_{ij}\tilde{\lambda}_j^k}. \quad (24)$$

This notation has the property that  $\sum_j \lambda_j^{k+1} = \sum_i p_i$ , in other words the sum of the approximation is equal to the number of registered hits for each approximation.

In matrix vector notation the relation becomes,

$$\lambda^{k+1} = \text{diag}(p^T \text{diag}^{-1}(M\lambda^k)M)\lambda_k. \quad (25)$$

To test this substitution we use a non smooth distribution of tracers as shown in figure 11(a). This distribution is on a 30 by 30 grid with 8 pinholes each with

120 pixels.

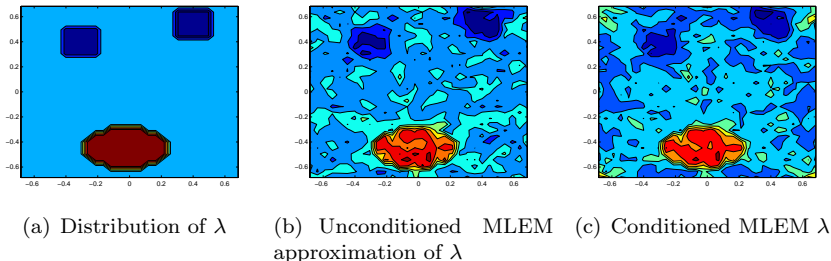


Figure 11: Standard and adjusted MLEM approximation of the distribution

The total amount of generated photons is in the order of  $1.0+9$  of which only a small fraction is registered. We notice artefacts in the image produced by formula 24 and despite efforts to reduce these by multiplying the outcome by  $D$  to some power other than 1, the image remained more noisy. We conclude using formula 24 is not a good idea.

### 5.3 Stopping Conditions

Choosing the correct stopping condition is hard as the norms of the equations are not necessarily related to the norm of the actual error. Various medical literature mentions the problem of choosing the stopping condition and points out that in practice a set amount of iterations is used.

The simulations used for this thesis use large, arbitrary stopping criteria as the purpose of this work is to compare the behaviour of several methods. We know the distribution of tracers so we simply use this to determine the fitness of each method by observing which approximation leads to the smallest error.

Another way to determine the accuracy of each algorithm the error is defined as the difference between the approximation and the best possible estimate of the unknown tracer distribution. The best estimate is to take the amount of detected, among all pixels, radiation for each grid element. Normally this is not known as the radiation is spread out among several pixels and adds to the detected photons from various other grid elements. Also this estimate suffers from Poisson random noise as it is determined directly from the sent radiation.

More recently the problem of choosing a good stopping condition was discussed by Gaitanis et al. in [4]. In this paper the authors look at the normalized root mean square deviation NRMSD and the chi-square between the approximation and the actual tracer distribution. Though these quantities are not known in practice analyses show that these quantities both reach an optimal minimum whereas the log likelihood monotonously increases. As the root mean square deviation and the chi-square are unknown in practice the authors look at the distribution of the update vector  $C^k$  which is defined as the vector which



is multiplied with the  $k$ th approximation to get the  $(k + 1)$ th MLEM approximation,

$$C_j^k = \frac{1}{\sum_i M_{ij}} \sum_i \frac{M_{ij} p_i}{\sum_j M_{ij} \lambda_j^{(k)}}, \quad \forall j. \quad (26)$$

Because the MLEM algorithm converges, we can conclude  $\lim_{k \rightarrow \infty} C_j^k \rightarrow 1 \quad \forall j$ .

Figure 12 shows  $C^k$  for 5 random coefficients of the update vector  $C$  for the first 120 iterations.

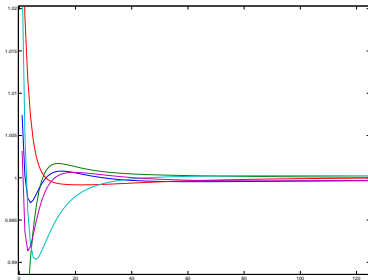


Figure 12: Convergence of the update vector. The X-axis shows the iterations, the Y-axis the update values.

The authors of [4] looked at the minimum of the coefficients  $C_j$  and noticed the optimal normalized root mean square deviation occurs at a specific value of the minimal coefficient. This value only seems to depend on the number of registered counts and not on the actual geometry of the solution nor on the discretization. Unfortunately the effect of scattering and attenuation is not looked into but if the observations are true in general or for a broad enough space of the possible tracer distributions, this result justifies taking a fixed number of iterations in practice.

#### 5.4 Effects of the number of projection points

We saw in Section 3.2 that the number of projection points have a dramatic effect on the condition number of the matrix  $MM^*$ . To see the effect of increasing the number of projection points on the norm of the error. A non smooth distribution of tracers was approximated with LS, the result is displayed in figure 13, the y-axis shows the norm of the difference between  $\lambda$  and its approximation. A grid size is 32 by 32 was used with the number of projection points is 8 by 128 (red), 8 by 140 (blue) and 8 by 256 (green). Each configuration was run three times to lessen the effect of statistical deviants. In this picture we see the approximations tend to become better with more projections points.

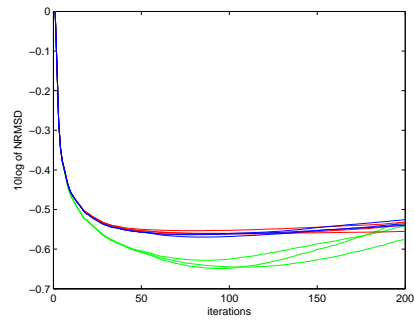


Figure 13: The norm of the error for several amounts of projection points

The next picture, see 14 displays the same configuration as the previous with the difference that the source distribution is smooth. It shows a different behavior of the error when the number of projections points is increased.

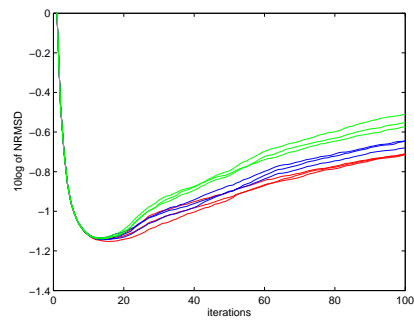


Figure 14: The norm of the error for several amounts of projection points

When zoomed in, see figure 15, the error even increases with the number of projections points though only marginally. Also note that the better approximations tend to require more iteration steps.

The source distribution was chosen to be irregular.

It is not possible to conclude that decreasing the condition number of the matrix will lead to better approximations or faster convergence. Better approximations do tend to require more iterations steps, though this effect is only marginally. A possible factor might be that as the number of projection points increases the information is spread out over more bins making the reconstruction of the original distribution harder.

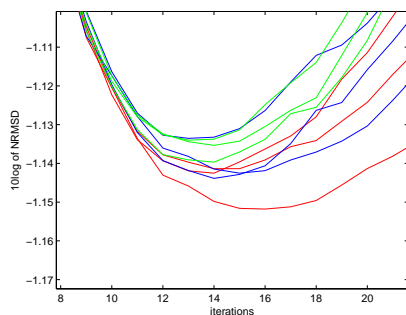


Figure 15: Zoomed view of figure 14

## 5.5 Performance of MLEM

Using a smooth test function as defined in 22 we focus in this paragraph on the performance of the MLEM algorithm using different amounts of radiation. The simulation consists of a grid size of 32 by 32 with 8 pinholes each with a resolution of 128.

During a scan the patient or animal needs to remain still thus one prefers a short scanning time which leads to a low photon count. The best algorithm will therefore needs to perform well with a low amount of registered photons.

Figure 16 shows the convergence of MLEM for various amounts of expected radiation,  $1.0e7$  in red,  $1.0e8$  in blue and  $1.0e9$  in green. As expected a higher radiation count leads to a better approximation.

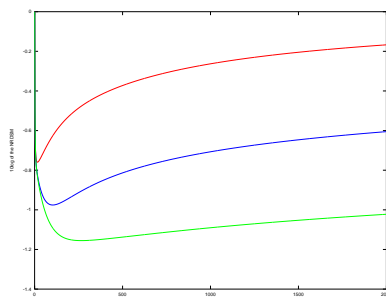


Figure 16: Convergence of MLEM for different amounts of photons. The X-axis shows the iterations, the Y-axis the NRMSD.

Unfortunately the number of needed iterations also increases. It turns out that for (unrealistically) high amounts of radiation the steps needed to reach the best approximation increases to a point the computation time becomes unworkable.

## 5.6 The relation between grid size and photon count

Increasing the grid size will decrease the inherent discretization error, more points are being approximated. Unfortunately the projection points will have to increase resulting in less photons for each projection bin. Figure 17 shows this effect for 64 (red), 256 (blue) and 1024 (green) grid points. Each configuration is shown with  $10^6$ ,  $10^7$ ,  $10^8$  and  $10^9$  decays. Each run with a higher amount of decays reaches a lower error. We notice the setup with 256 grid points and  $10^7$

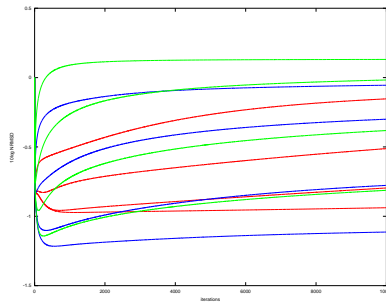


Figure 17: Convergence of MLEM for different grid sizes and photon counts

decays performs worse than the setup with 64 grid points and the same amount of radioactivity. At  $10^8$  decays however the setup with 256 grid points performs better than the setup with 64 grid points. We conclude a higher accuracy needs a higher amount of radioactivity.

## 5.7 Comparison between MLEM and LS

Using test function shown in section 5.5 we look at the performance of LS to compare it with the MLEM algorithm. Figure 18 shows the convergence of LS for different amounts of radiation.

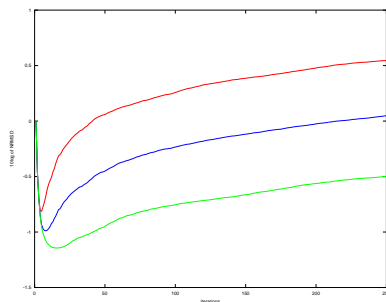


Figure 18: Convergence LS for different amounts of photons

Again the green line is the convergence plot with  $10^9$  decays, blue for  $10^8$  and red for  $10^7$ . Just like the MLEM algorithm the approximations become better as the amount of photons increases. The best approximations are reached in fewer iterations and with similar results. This behavior can be explained by the fact that a larger amount of photons leads to relatively smaller error. The largest eigenvalues are found first with CG, when the error is larger then the eigenvalue the approximation will drift away from the solution as the error is blown up.

To further compare the two methods a different the test function displayed in figure 11(a) and a total amount of expected decays of  $10^7$  was used. Figure 19 shows the convergence for both the MLEM (red) and the LS (blue) iterations both on the same system. We see a big difference with the previous comparison with the only change being the distribution of radiation source.

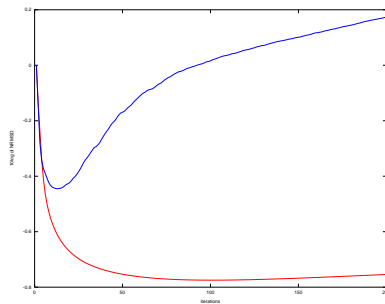


Figure 19: Convergence LS and MLEM

Figure 20 shows the approximation at the optimal number of iteration with  $1.0e7$  photons for both MLEM and LS.

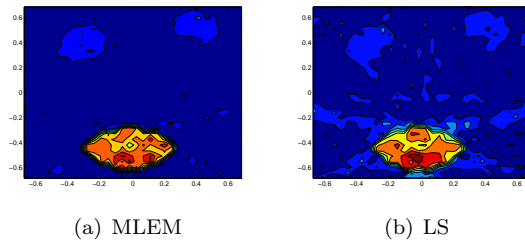


Figure 20: Approximation from MLEM and LS at the optimal number of iterations

The figure shows that the LS algorithm suffers more from artefacts. Also the approximation from LS has some negative values, this becomes more prominent as the background values are closer to zero.

## 5.8 Brute Force

In the previous sections the matrix  $M$  was square. Using an observed photon count equal to the number of expected photon count enables us to directly calculate an estimate for  $\lambda$  by multiplying the mean photon count by the pseudo-inverse of  $M$ . The matrix  $M$  is still singular. Figure 21 shows the solution, it is apparent this solution consists of only noise.

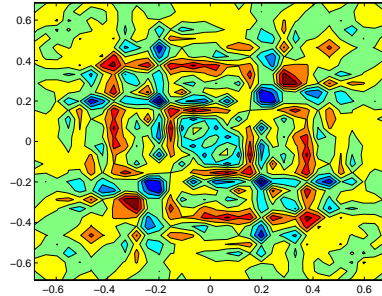


Figure 21: Straightforward calculation of  $\lambda$  with exact measurement and square matrix

## 5.9 The effect of substitution on the convergence of CGLS

To test the effect of using least squares combined with substitution in equation 9 a test was performed on a grid size of 32 by 32 using an expected number of  $10e7$  photons and 8 pinholes each with 256 projection bins. Figure 22 shows that the substitution had no effect. The first 15 iterations were with standard CGLS on the system  $M'M\lambda = M'p$ . The following 15 were also with CGLS but now on the system  $M'_\lambda M_\lambda \lambda = M'p_\lambda$  with  $M_\lambda = \text{diag}(M\lambda)^{-1/2}M$  and  $p_\lambda = \text{diag}(M\lambda)^{-1/2}p$ .

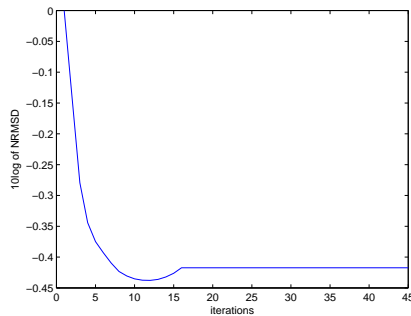


Figure 22: Convergence plot for substitution

## 5.10 performance of the Newton-Raphson method

To test the convergence of the Newton-Raphson method the exact same configuration as in section 5.9 was used. The result was rather disappointing with a decline in error only in the first LS step, the next step where  $\delta\lambda$  is calculated immediately caused the solution to become not a number.

## 5.11 Effect of Radiation

An increase in the amount of radiation tends to lower the error as the measured data becomes more accurate. Figure 23 shows this effect. Starting from the top each line shows the convergence with a ten fold increase in radiation. The figure suggests accuracy increases in the order of the square root of the radiation. Also at lower radiations substituting  $\lambda$  has no effect until at some radiation increases the radiation has no effect on the accuracy of the first step, at that point substituting starts to increase the accuracy.

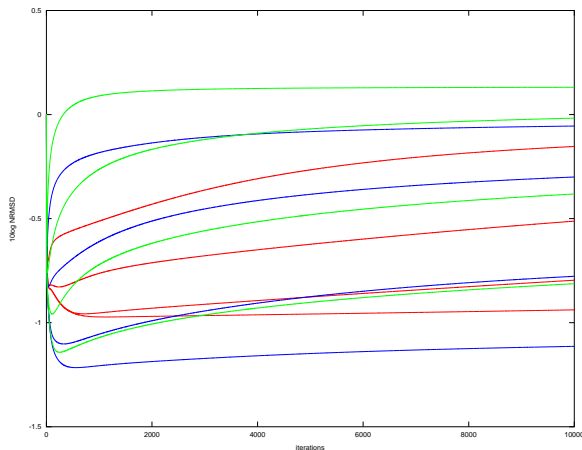


Figure 23: Convergence for different radiations

## 5.12 Convergence for different foci

The 3D U-Spect scanner was designed to only allow a small range of angles to pass the pinholes. By narrowing the scope of these pinholes the resulting matrix will be more decoupled. To test the effect of varying the viewing angle of the pinholes the convergence of MLEM was tested with angles of  $\pi/2$ ,  $\pi/4$  and  $\pi/8$ , respectively red, blue and green in figure 24.

For this test a grid size of 16 by 16 was used with 8 pinholes each with 32 projection bins. For this set-up lessening the viewing angle has a similar effect as decreasing the radioactivity, the best approximation becomes worse.

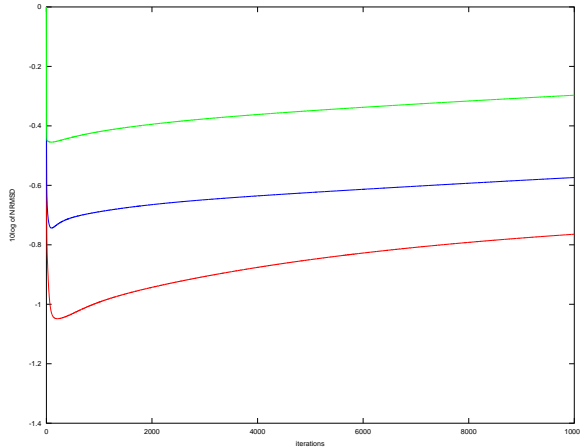


Figure 24: Convergence for different viewing angles

## 6 Conclusion

The goal for this thesis was to develop a better method to solve emission based image reconstruction than the standard MLEM algorithm. CGLS showed a good speed increase and a reasonable error reduction though MLEM outperforms CGLS on that regard. The idea was then to solve the MLEM representation, equation 8 or 9, directly. Both two methods one using substitution to update the matrices another using Newton-Raphson fail to converge. We conclude that though MLEM can be slow its higher error reduction makes it more suitable than the analyzed methods, unless the radiation levels are extremely high. Further research could focus on solving the MLEM equation 8 combined with the constraint that  $\lambda > 0$ .

## A Matlab source code

### A.1 Main file

To generate the Poisson distributed variables the matlab routine Randraw was used.

```
clear all
angles=[-pi+pi/4:pi/4 :pi]; %angles of pinholes, in radians from -pi to pi
para(1)=1 ; %radius of the tube
para(2)=2 ; %length of the projection grid
para(3)=256; %number of pixels for each pinhole
para(4)=32; %number of gridpoints in both x and y direction
```



```

para(5)=1;      %size of square grid,
%if square is smaller then the radius of the tube then some pixels will never be hit by radiation
% which causes some rows in the matrix to remain zero.
para(6)=1.0e+7; %equals the total amount of atoms that will decay during the test, because
%statically the same fraction of the tracers will decay we can assume para(6) depends linearly
% on the amount of tracers. The amount of decayed tracers also depends on the amount of time but
%as decay is exponentially this dependency is non linear. A typical dose (for PET scans) is 400Mbg
%which is 400 million decays per second, a scan can take up to 15 minutes.
para(7)=pi/200; %width of the pinholes in radians
para(8)=2; % parameter to decide the distribution of the tracers
para(9)=pi/2; %the width of "vision" in radians for each pinhole

numberofpixels= para(3)*numel(angles);
numberofgridpoints= para(4)^2;
gridelement_area=(para(5)/para(4))^2;
use_best_estimate=0;
%setting this true will calculate the measured valued as a sum of several Poisson distributed
%variables.This makes the best estimator available which gives a more fair comparison.
fprintf('Computing A\n')
tic
A=constructA(angles, para);
toc
rand('state', 57); %some random seed

exact_solution=constructx(para);

fprintf('Computing b\n')
tic
b=zeros(numberofpixels,1);
bmean=A*exact_solution;

optimal_estimate=zeros(numberofgridpoints,1);
if use_best_estimate==0
%this is a great deal faster then computing each contribution as a possion distributed variable
for i=1:numberofpixels
    b(i)=randdraw('po',bmean(i),[1 1]);
end
end

if use_best_estimate==1
for i=1:numberofpixels
for j=1:numberofgridpoints
    if A(i,j)>0
var=randdraw('po',A(i,j)*exact_solution(j),[1 1]);
b(i)=b(i)+var;
optimal_estimate(j)= optimal_estimate(j)+var;

```

```

        end
    end
end
end
optimal_estimate= optimal_estimate./sum(A)';
toc

%the following parameters are used to call various methods to compute an approximation
%0=do not use the method, 1=use the method, 2=use the method and plot a convergence plot.
MLEM=0;
LS=0;
LSSUB=0;
NR=2;
NLCG=0;
CG=0;
SCG=0;
SCG2=0;

if MLEM>0
    tic
    fprintf('Using the MLEM method\n')
    lambda_MLEM=zeros(numberofpixels,1);
    lambda_MLEM=MLEM_Method2(A,b,optimal_estimate,exact_solution,para,MLEM);
    toc
end

if LS>0
    tic
    fprintf('Using the LSCG method\n')
    lambda_LS=zeros(numberofpixels,1);
    lambda_LS=LS_Method(A,b,optimal_estimate,exact_solution,para,LS);
    toc
end

if LSSUB>0
    tic
    fprintf('Using LSCG and substitution to solve the MLEM equations\n')
    lambda_LSSUB=zeros(numberofpixels,1);
    lambda_LSSUB=LSSUB_Method2(A,b,optimal_estimate,exact_solution,para,LSSUB);
    toc
end

if NR>0
    tic
    fprintf('Using Newton-Raphson to solve the MLEM equations\n')
    lambda_NR=zeros(numberofpixels,1);
    lambda_NR=NR_Method(A,b,optimal_estimate,exact_solution,para,NR);
    toc
end

```

```

end

if NLCG>0
    tic
    fprintf('Using Non Linear CG to solve the MLEM equations\n')
    lambda_NLCG=zeros(numberofpixels,1);
    lambda_NLCG=NLCG_Method(A,b,exact_solution,para,NLCG);
    toc
end

```

## A.2 Constructing the matrix M

```

function varargout = constructA(angles, para)

radius1 =para(1);
radius2 =para(2);
numberofpixels= para(3);
numberofgridpoints= para(4);
sizeofgrid =para(5);
collimator_angle=para(7);
pinhole_vision=0.5*para(9);

ep=0.000000001;          %small value to deal with rounding errors
area_square=(sizeofgrid/numberofgridpoints)^2;
focus=zeros(numel(angles),2);
linedirections=zeros(numberofpixels+1,2);
%each collimator has as much lines associated as the number of pixels plus 1
angle_right=zeros(numel(angles),1);

A=sparse(numel(angles)*numberofpixels,numberofgridpoints^2);

for j=1:numel(angles) %for each collimator calculate the focus point and the reference line angle
    %location of the collimator and starting point for each line passing it
    focus(j,:)=[radius1*cos(angles(j)) radius1*sin(angles(j))];
    tangle=angles(j)-pinhole_vision; %the lowest angle passing collimator j
    tempright=[cos(tangle) sin(tangle)]; %vector with length 1
    b=2*tempright(1)*focus(j,1)+2*tempright(2)*focus(j,2);
    c=radius1^2-radius2^2;
    t=-b/2+sqrt(b^2-4*c)/2; %the quadratic formula, note that a = 1
    rightpoint=focus(j,,:)+t*tempright;
    angle_right(j)=atan2(rightpoint(2),rightpoint(1)); % angle_right in <-pi,pi]
end

for j=1:numel(angles) %for each collimator
    for i=0:numberofpixels %for each line marking the projection region of a pixel

```

```

    angle_line=angle_right(j)+2*i*(angles(j)-angle_right(j))/numberofpixels;
%total width of the projection is twice of angle between the utter right ray and the middle
    if angles(j)<0 && angle_right(j)>0 %this is when the angle wraps around
        angle_line=angle_right(j)+2*i*(angles(j)+2*pi-angle_right(j))/numberofpixels;
    end
    anglepoint=radius2*[cos(angle_line) sin(angle_line)];
    linedirection = [focus(j,1)-anglepoint(1) focus(j,2)-anglepoint(2)];
%direction of the line through collimator j
    for k=0:numberofgridpoints
        s=-sizeofgrid/2+k*sizeofgrid/numberofgridpoints;
        if linedirection(1)==0
%if the line is vertical intersection with vertical grid lines are set at a
%focus point which lies outside the grid
            intersections(2*k+1)=0;
        else
            intersections(2*k+1)=(s-focus(j,1))/linedirection(1);
%setting the elements equal to the intersections with the x and y lines respectively
        end
        if linedirection(2)==0
            intersections(2*k+2)=0;
        else
            intersections(2*k+2)=(s-focus(j,2))/linedirection(2);
        end
    end
    intersections=sort(intersections);
%sorting intersection such that intersections within the region of interest are sorted
    for k=1:numel(intersections)-1 %for each bisection of a square
        inter1=focus(j,:)+intersections(k)*linedirection;
        if (inter1<=[sizeofgrid/2,sizeofgrid/2]+ep & inter1>=-[sizeofgrid/2,sizeofgrid/2]-ep)
%if first intersection is within or on edge of grid
            inter2=focus(j,:)+intersections(k+1)*linedirection;
            if (inter2<=[sizeofgrid/2,sizeofgrid/2]+ep & inter2>=-[sizeofgrid/2,sizeofgrid/2]-ep)
%and second intersection in within or on edge of grid
%[inter1 inter2] these two intersections bisect a grid square
                point=(inter1+inter2)/2;
% point is in the interior of a square unless a line goes through the edge of the square
                point=numberofgridpoints*(point+[sizeofgrid sizeofgrid]/2)/sizeofgrid;
                gridindex=fix(point); %the index within the grid of the bisected square
                middle=[-sizeofgrid/2+(gridindex(1)+0.5)*sizeofgrid/numberofgridpoints -
                    sizeofgrid/2+(gridindex(2)+0.5)*sizeofgrid/numberofgridpoints];
%middle of the bisected square
                x=1;
                if (abs(inter1(1)-inter2(1))>sizeofgrid/numberofgridpoints-ep)
% in case of a horizontal bisection
                    x=0;
                    a=inter1(2)-gridindex(2)*sizeofgrid/numberofgridpoints+0.5*sizeofgrid;

```

```

b=inter2(2)-gridindex(2)*sizeofgrid/numberofgridpoints+0.5*sizeofgrid;
area=0.5*(a+b)*sizeofgrid/numberofgridpoints;
if (inter1(1)>inter2(1))
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)>0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=
            A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)-area;
    end
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)==0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area_square-area;
    end
    if(i<numberofpixels)
        A((j-1)*numberofpixels+i+1,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area;
    end
end
if (inter2(1)>inter1(1))
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)>0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=
            A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)-area_square+area;
    end
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)==0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area;
    end
    if(i<numberofpixels)
        A((j-1)*numberofpixels+i+1,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area_square-area;
    end
end
end
if ((abs(inter1(2)-inter2(2))>sizeofgrid/numberofgridpoints-ep) & x==1)
% in case of a vertical bisection
x=0;
a=inter1(1)-gridindex(1)*sizeofgrid/numberofgridpoints+0.5*sizeofgrid;
b=inter2(1)-gridindex(1)*sizeofgrid/numberofgridpoints+0.5*sizeofgrid;
area=0.5*(a+b)*sizeofgrid/numberofgridpoints;
if (inter1(2)<inter2(2))
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)>0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=
            A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)-area;
    end
    if(i>0 && A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)==0)
        A((j-1)*numberofpixels+i,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area_square-area;
    end
    if(i<numberofpixels)
        A((j-1)*numberofpixels+i+1,gridindex(2)*numberofgridpoints+gridindex(1)+1)=area;
    end
end
end
if (inter2(2)<inter1(2))

```



```

end

for i=1: numel(angles)
%for each pinhole, this part is mainly to project grid elements who have
%not been intersected by lines to pixels
x= radius1*[cos(angles(i)+collimator_angle/2) sin(angles(i)+collimator_angle/2)];
%x and y are the locations of each end of a pinhole
y= radius1*[cos(angles(i)-collimator_angle/2) sin(angles(i)-collimator_angle/2)];
for j=1:numberofgridpoints^2 %for each grid element
gridindex(1)=mod(j-1,numberofgridpoints); %both gridindexes are in [0,numberofgridpoints>
gridindex(2)=fix((j-1)/numberofgridpoints);
middle=[-sizeofgrid/2+(gridindex(1)+0.5)*sizeofgrid/numberofgridpoints -
sizeofgrid/2+(gridindex(2)+0.5)*sizeofgrid/numberofgridpoints];
m=[focus(i,1)-middle(1) focus(i,2)-middle(2)];
a=m(1)^2+m(2)^2;
b=2*m(1)*focus(i,1)+2*m(2)*focus(i,2);
c=radius1^2-radius2^2;
t=(-b+sqrt(b^2-4*c*a))/(2*a); %the quadratic formula
middleprojection=focus(i,:)+t*m;
angle_middle=atan2(middleprojection(2),middleprojection(1)); % angle in <-pi,pi]
if angles(i)<angle_right(i)
%if the angle wraps around pi/-pi, remember angle_right should be lower then angle
angles(i)=angles(i)+2*pi;
end
if angles(i)>0 && angle_right(i)>0 && angle_middle<0
angle_middle=angle_middle+2*pi;
%if projectionline of middle wraps around pi/-pi, or a degenerated case when angle and
%angleright are close to zero and angle_middle fall outside the projector by being negative
end
if angle_middle>angle_right(i) && angle_middle<2*angles(i)-angle_right(i)
%if middle of element projects into a pixel
tempindex=numberofpixels*(i-1)+
fix(numberofpixels*((angle_middle-angle_right(i))/(2*(angles(i)-angle_right(i)))))+1;
%index of the pixel on which the element is projected
if A(tempindex,j)==0 %if corresponding element has not been set yet
A(tempindex,j)=area_square; %project the entire area of the element into the pixel
end
end
u=x-middle; %we adjust the pinhole ends x and y relative to the middle of a grid element
v=y-middle;
rel_pinhole_angle=acos((u(1)*v(1)+u(2)*v(2))/(norm(u)*norm(v)));
%known sides so we can calculate the angle using the cosine law
A(numberofpixels*(i-1)+1:numberofpixels*(i-1)+numberofpixels,j)=
rel_pinhole_angle/(2*pi)*A(numberofpixels*(i-1)+1:numberofpixels*(i-1)+numberofpixels,j);
%adjusts matrix elements to the relative angle of the pinhole under the assumption radiation is
%distributed evenly in each direction

```

```

end
end
varargout{1} = A;
end

```

### A.3 Constructing the tracer distribution

```

function varargout = constructx(para)
distribution=para(8);
sizeofgrid= para(5)/2;
numberofgridelements=para(4);
stepsize=para(5)/numberofgridelements;
Xmesh=-sizeofgrid+stepsize/2:stepsize:sizeofgrid-stepsize/2;
Ymesh=Xmesh;
x=ones(numberofgridelements^2,1);

if distribution==1
%a sample smooth distribution
for i=1:numberofgridelements
    for j=1:numberofgridelements
        x((i-1)*numberofgridelements+j)=cos(2*pi*Xmesh(i))*cos(2*pi*Ymesh(j))+1;
    end
end
end

if distribution==2
% a more realistic non-smooth distribution
disx1=0.5*numberofgridelements;
disy1=0.2*numberofgridelements;
disx2=0.3*numberofgridelements;
disy2=0.8*numberofgridelements;
disx3=0.8*numberofgridelements;
disy3=0.9*numberofgridelements;
for i=1:numberofgridelements
    for j=1:numberofgridelements
        x((i-1)*numberofgridelements+j)=0.1;
        if sqrt((i-disx1)*(i-disx1)+3*(j-disy1)*(j-disy1))<0.2*numberofgridelements
            x((i-1)*numberofgridelements+j)=2;
        end
        if sqrt((i-disx2)*(i-disx2)+(j-disy2)*(j-disy2))<0.1*numberofgridelements
            x((i-1)*numberofgridelements+j)=0.5;
        end
        if sqrt((i-disx3)*(i-disx3)+(j-disy3)*(j-disy3))<0.1*numberofgridelements
            x((i-1)*numberofgridelements+j)=0.4;
        end
    end
end
end

```



```
end
end
```

```
if distribution==3
%the following lines read a color picture into 3 matrices which can be reconstructed separately.
filesize=50;
IMAGE8=imread('1044463682_small-image_pmonblackblue1921sm_edit','bmp');
IMAGE64=double(IMAGE8)/255;
RED=IMAGE64(1:filesize,1:filesize,1);
Vred=RED(:);
GREEN=IMAGE64(1:filesize,1:filesize,2);
Vgreen=GREEN(:);
BLUE=IMAGE64(1:filesize,1:filesize,3);
Vblue=BLUE(:);
x=Vred;
end
```

```
x=x*para(6)/(sum(x)*stepsize*stepsize);
%x is normalized and multiplied by the amount of expected particle emissions
varargout{1} = x;
end
```

#### A.4 The MLEM method

```
function [lambda]=MLEM_Method2(A,b,optimal_solution,exact_solution,para,showplot)
dimA=size(A);

lambda=ones(dimA(2),1)*sum(b)/dimA(2); %initial lambda is the average number of hits
columnsum=zeros(dimA(2),1);
for j=1:dimA(2)
    columnsum(j)=sum(A(1:dimA(1),j));
end
columnsum=columnsum.^-1;
i=1;
normx=norm(optimal_solution);
if showplot==2
    NRMSD(1)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
end

iter=10000;
while i<iter
    xi=(A*lambda).^-1;
    pit=b.*xi;
    lambda_new=lambda.*columnsum.*(A'*pit);
```

```

lambda=lambda_new;
i=i+1;
if showplot==2
    NRMSD(i)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
end
end
if showplot==2
    figure(1)
    plot(log10(NRMSD),'r');
end
end
end

```

## A.5 The LS method

```

function [lambda]=LS_Method(A,b,optimal_estimate,exact_solution,para,showplot)
%this function solves A'Ax=A'b using CG
dimA=size(A);
lambda=zeros(dimA(2),1);
Z=lambda;
newb=A'*b;
r=newb;
p=r;
Ap=zeros(dimA(2),1);
residue=1;
i=1;
normx=norm(exact_solution);
iter=15;
for i=1:iter
    if showplot>1
        NRMSD(i)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
    end
    Ap=A*p;
    alpha=r'*r/(Ap'*Ap);
    lambda=lambda+alpha*p;
    r2=r-alpha*A'*Ap;
    beta=r2'*r2/(r'*r);
    p=r2+beta*p;
    r=r2;
end
if showplot>1
    figure(1)
    plot(log10(NRMSD),'b');
    NRMSD
end
end

```

## A.6 The LS method with substitution

```
function[lambda]=LSSUB_Method(A,b,optimal_solution,exact_solution,para,showplot)
%This function uses LSCG to get the initial aproximation for lambda then substitutes the
% approximation into the non linear function  $M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot M \cdot \lambda = M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot p$ 
% a number of times to solve this equation.
dimA=size(A);
lambda=zeros(dimA(2),1);
LS=2;
newA=A;
newb=b;
iter=4;
normx=norm(exact_solution);
if showplot==2
    NRMSD(1)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
end
normx=norm(exact_solution);
for k=2:iter
    iterationstep=k
    lambda=LS_Method(newA,newb,optimal_solution,exact_solution,para,LS);
    Ax=A*lambda;
    Ax=max(Ax,eps);
    Ax=Ax.^(-1/2);
    newA=diag(Ax)*A;
    newb=diag(Ax)*b;
%substitute into equation  $M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot M \cdot \lambda = M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot p$  (see thesis)
%to get  $\text{newA}' \cdot \text{newA} \cdot x = \text{newA}' \cdot \text{newb}$  which can be approximated by LSCG
    if showplot==2
        NRMSD(k)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
    end
end

if showplot==2
    figure
    plot(log10(NRMSD),'b');
    NRMSD
end
```

## A.7 The Newton Raphon method

```
function[lambda]=NR_Method(A,b,optimal_estimate,exact_solution,para,showplot)
%This function uses LSCG to get the initial aproximation for lambda then substitutes the
%approximation into the non linear function  $M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot M \cdot \lambda = M' \cdot \text{diag}(M \cdot \lambda)^{-1} \cdot p$ 
%a number of times to solve this equation.
dimA=size(A);
numberofgridpoints=dimA(2);
```

```

LS=1;
M=A;
lambda=zeros(numberofgridpoints,1);
p=b;
normx=norm(exact_solution);
for k=1:10
    if showplot==2
        NRMSD(k)=sqrt(sum((lambda-exact_solution).^2)/sum(exact_solution.^2));
    end
    delta=LS_Method(M,p,optimal_estimate,exact_solution,para,LS);
    lambda=lambda+delta;
    M=diag((A*lambda).^(-1)*diag((b.^(1/2))))*A; %grad F = M*M
    p=b.^(1/2)-diag(A*lambda)*b.^(-1/2);

end

if showplot==2
    figure
    subplot(1,1,1), semilogy(NRMSD), title('NRMSD');
    NRMSD
end

```

## References

- [1] A. M. Ali, Z. Melegyd, M. Morsyd, R. M. Megahida, T. Bucherlb, and E. H. Lehmann. Image reconstruction techniques using projection data from transmission method. *Annals of Nuclear Energy*, 31(12):1415–1428, 2004.
- [2] I. Csiszar and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics and Decisions, Supplement Issue*, 1(1):205–237, 1984.
- [3] A. P. Dempster, N.M Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.
- [4] A. Gaitanis, G. Kontaxakis, G. Spyrou, G. Panayiotakis, and G Tzanakos. Pet image reconstruction: A stopping rule for the mlem algorithm based on properties of the updating coefficients. *Computerized Medical Imaging and Graphics*, 34(2):131–141, 2010.
- [5] William W. Hager and Honhchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2 (2006), pp. 35-58., 2006.
- [6] H. M. Hudson and R. S. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Transactions on medical imaging*, 13(4):601–609, 1994.
- [7] E. A. Nederkoorn. Richardson-lucy fitting for u-spect calibration. Master’s thesis, Universiteit Utrecht, 2006.
- [8] J.B.T.M. Roerdink. *Encyclopaedia of Mathematics*, volume Tomography. Springer, <http://eom.springer.de/t/t092980.htm>, 2001.
- [9] L.A Shepp and Y. Vardi. maximum likelihood reconstruction in positron emission tomography. *IEEE Transactions on medical imaging*, 1(2):113–122, 1982.
- [10] F. van der Have, B. Vastenhouw, R. M. Ramakers, W. Branderhorst, J. O. Krahe, C. Ji1, S. G. Staelens, and F. J. Beekman. U-spect-ii: An ultra-high-resolution device for molecular small-animal imaging. *The Journal of Nuclear Medicine*, 50(4):599–605, 2009.