UNIVERSITÄT ZU LÜBECK

Bachelor Thesis

An augmented Lagrangian method for inequality constrained optimization applied to SPECT reconstruction

submitted by
Johannes Lötscher

supervised by
Prof. Dr. rer. nat. Jan Modersitzki
Institute of Mathematics and Image Computing

and
Dipl. Inf. Sven Barendt
Institute of Mathematics and Image Computing

June 6, 2011

Statement of authorship

I hereby certify that this thesis has been composed by me and is based on my own work, unless stated otherwise. This work has not been submitted for any other degree.

Lübeck, June 6, 2011 Johannes Lötscher

Abstract

In medical imaging there are a lot of ill-posed problems, as for example SPECT Reconstruction. A way to deal with such problems is optimization. A crucial part of optimization is to choose a appropriate model for the problem. This is the point where the application of constraints can be handy. The *augmented Lagrangian method* [4] provides a strategy to handle equality and inequality constraints by introducing the augmented Lagrangian function. The aim is now to find a minimum in this function, which is accomplished by a Newton-like method. For this thesis I implemented an algorithm in MATLAB which solves this minimization problem. The implementation is tested with some small examples, that are easy to understand and visualize to explain how the algorithm works. But in order to really see the influence of a more sophisticated model, the method is applied to SPECT Reconstruction. I worked with three optimization models, the first one is an unconstrained model, where I simply try to minimize the the sum of squared differences of the SPECT Measurement and the reconstructed Image. For the second model I considered that the activity of a SPECT Reconstruction is always greater or equal zero. The third model tries to minimize the overall activity of the reconstructed image with the nonnegativity constraints and a restriction the the sum of squared differences.

The evaluation of these three models show that the first two are provide similar reconstruction results. They needed about 50 iterations to reach a result that comes near the original image. But the image is still quite blurry and there are unwanted artifacts. The third model provides much better results. Already after 10 iterations the image looks very similar to the original, with sharp edges and less artifacts.

Zusammenfassung

In der medizinischen Bildverarbeitung gibt es viele schlecht gestellte Probleme, darunter zum Beispiel die SPECT Rekonstruktion. Eine Strategie solche Probleme zu lösen ist Optimierung. Ein entscheidender Schritt bei der Optimierung besteht darin, ein entsprechendes mathematisches Modell zu wählen, dabei kann die Verwendung von Nebenbedingungen vorteilhaft sein. Das Augmented-Lagrange-Verfahren [4] liefert eine Strategie, die es ermöglicht Gleichheits- und Ungleichheitsnebenbedingungen durch die Einführung der Augmented-Lagrange-Funktion zu verwenden. Das Ziel ist es in dieser Funktion mit Hilfe eines Newton-Verfahrens ein Minimum zu finden. Für diese Arbeit habe ich in MATLAB einen Algorithmus implementiert, der dieses Minimierungsproblem löst. Die Implementierung wurde an kleinen, gut verständlichen und einfach visualisierbaren Beispielen getestet und untersucht. Um den Vorteil eines besser gewählten Optimierungsmodells sichtbar zu machen wurde die Methode auf SPECT Rekonstruktion angewendet. Dazu wurden drei verschiedene Modelle zur Rekonstruktion aufgestellt und verglichen. Das erste Modell versucht den Fehler bezüglich eine Summe von quadrierten Differenzen (ein Distanzmaß) ohne Nebenbedingungen zu minimieren, das zweite verwendet die Selbe Zielfunktion soll diese jedoch minimieren ohne dass negative Aktivitäten erlaubt sind, das dritte Modell verwendet als Zielfunktion die Summe aller Aktivitäten und hat sowohl die Nebenbedingung, die keine negativen Aktivitäten zulässt, als auch eine Restriktion bezüglich des Distanzmaßes.

Die Evaluation dieser drei Modelle zeigt, dass sich die ersten beiden Modelle kaum unterscheiden. Nach etwa 50 Iterationen kommt die Lösung dem Original relativ nahe, allerdings ist das Bild leicht verschwommen und es zeigen sich Artefakte ausserhalb des interessanten Bereichs. Bei dem dritten Modell kommt es bereits nach 10 Schritten zu einem sehr zufriedenstellenden Ergebnis. Die Kanten sind scharf, wie bei dem Original und es zeigen sich keine Artefakte bis auf ein Rauschen.

Contents

1 Introduction

1.1 Introducing constraints

In the field of medical image computing there are a lot of inverse problems, which are charactarized by the fact that there is a measurement and based on this data the causing physical parameters have to be obtained. In the context of computed tomography this means that there is a measurement of radiation and now the attenuation map in case of CT or the acitivity distribution of some radiopharmaceutical in case of SPECT has to be computed.

Inverse problems are difficult to solve, because normally there is no straight-forward strategy that provides a satisfiable solution. An approach to handle such problems is optimization, where the best result concerning the mathematical model has to be found. So the choice of an appropriate model is crucial to ensure that the optimal solution of the model is comes as close as possible to the optimal solution in reality. An important point about modeling in optimization is to define a suitable and well conditioned objective function. But even if the chosen function keeps this conditions it is possible that an algorithm computes a solution that is not feasible. This encourages to use some kind of restrictions to the solution, which define a set of solutions which are feasible. This leads to constrained optimization, it allows to enhance the model by adding constraint functions to the objective function.

1.2 Optimization in SPECT

SPECT is a functional medical imaging method, that allows the visualisation of metabolic functions. The technique uses a radioactive marker, which is injected into a patient. Depending on which marker is used, it is bound to different kinds of tissue. The marker is emitting gamma rays which then are measured by an array of sensors rotating around the patient. The sensor data provides 2-dimensional projections of the radiation. The task, which is the issue in this thesis, is to compute the 3-dimensional activity distribution, which corresponds to the distribution of the marker in the body of the patient.

The difficulty in the reconstruction the activity distribution f arises from the fact, that the radiation measured by the sensor is attenuated by the tissue between the radiating marker and the sensor. So in order to reconstruct the f an algorithm needs information

about the tissue density μ. The problem is that in practice this information is not known.

1.3 Structure of thesis

This thesis starts with an introduction to the theoretical background of the optimization methods that will be used. It begins with the simple cases for unconstrained optimization and advances to equality constrained optimization, which is explained to prepare the final point of interest: inequality constrained optimization.

The next part will deal with the actual implementations of the methods. The algorithms will be explained and evaluated for small examples that are easy to understand and visualize.

In the last part the algorithms will be applied to SPECT reconstruction. In order to make a statement about the advantages of a more sophisticated optimization model, two algorithms will be compared. The first one will have simple model and tries to reconstruct by unconstrained optimization, the second one will use the advanced model and uses an inequality constrained optimization algorithm based on the augmented Lagrangian method.

2 Optimization theory

In this section a theoretical background for methods to solve optimization problems will be provided. The concepts are based on the excellent book of Nocedal and Wright [4]. The knowledge about these concepts will allow it to implement practical algorithms. Several optimization methods for different kinds of problems will be presented and some difficulties will be discussed which can occur when applying these methods.

Optimization problems can be classified into unconstrained, equality-constrained and inequality-constrained optimization. The general optimization problem can be formulated as

$$\min_x \ f(x) \tag{2.1}$$
$$\text{s.t. } c_i(x) = 0, \ i \in \mathcal{E} \tag{2.2}$$
$$c_i(x) \geq 0, \ i \in \mathcal{I}, \tag{2.3}$$

where f denotes the *objective function* with the property $f : \mathbb{R}^n \to \mathbb{R}$ and c_i are the *constraints* with $c_i : \mathbb{R}^n \to \mathbb{R}$. $c_i(x)$ with $i \in \mathcal{E}$ represent the equality constraints and $i \in \mathcal{I}$ represent the inequality constraints.

A general approach to find an optimizer is to apply *line search methods*. They operate iteratively, which means they start at an initial guess x_0 for the optimizer and at each iteration compute a step, which should lead to a better solution. The algorithm terminates if an optimizer x^* satisfies certain *optimality conditions*. The computation of a step consists of two parts: first obtaining a *search direction* p_k and second determinig a *step length* γ. This results in the formula for the next iterate:

$$x_{k+1} = x_k + \gamma p_k$$

Line search methods can be applied to unconstrained and constrained optimization problems but there are different strategies that realize the line search approach.

If the objective function is convex, an appropriate line search method will find the global solution. If it is not convex, it will probably just find the local minimum next to the initial guess. The problem of finding the global minimum, will not be covered any further.

2.1 Unconstrained optimization

In unconstrained optimization the minimum of the objective function f has to be found:

$$\min_x f(x)$$

The first thing will be to derive some conditions, which allow to decide wether a point is a minimum or not.

Optimality conditions

For a point x^* the first-order optimality condition demands that there is no direction where an improvement of f is possible. This is the case if the gradient of f is zero:

$$\nabla f(x^*) = 0 \tag{2.4}$$

But this condition is not sufficient to guarantee a minimum, because it could also be a maximum or a saddle point. To ensure a minimum a second-order condition is necessary. If f is twice continuous differentiable, the function has to be locally convex in x^*, in other words this condition demands that

$$\nabla^2 f(x^*) \text{ has to be positive definite.} \tag{2.5}$$

As we have seen the first-order condition is *necessary* but not *sufficient*, only in combination with the second-order condition we have a *sufficient* optimality condition.

Steepest-descent method

This method is the most basic one to compute a search direction. It simply uses the gradient of the function at the current iterate x_k, which points towards increasing values of f. Defining the negative gradient as the search direction will result in a save descent direction.

$$p_k = -\nabla f(x_k)$$

Example 2.1. A small example will show how this looks. The objective function has the conture of an ellipse.

$$\min_x f(x) = x_1^2 + 5x_2^2$$

The minimum is obviously at $x^* = (0, 0)^T$. The implemented version of the steepest descent method takes 10 iterations to find a point satisfyingly near to x^*.

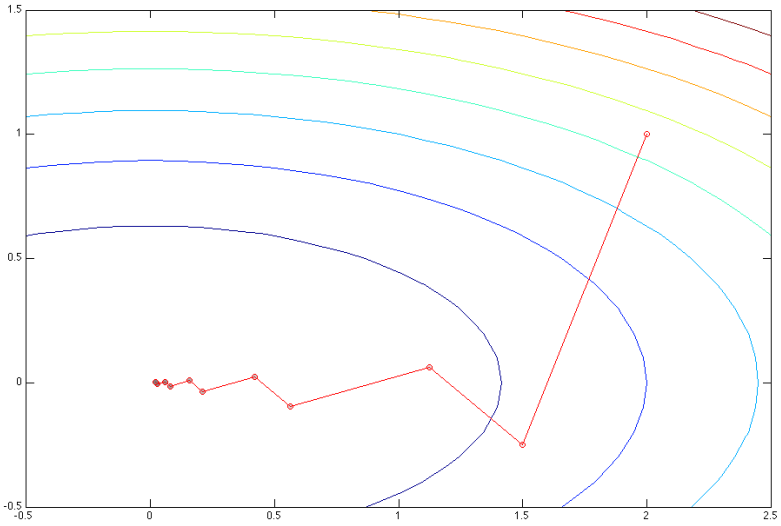


Figure 2.1: Example of an unconstrained optimization using the steepest descent method

As the figure shows, the search direction is always perpendicular to the conture lines. This can lead to very inefficient search direction as a modified example shows, with $\hat{f}(x) = x_1^2 + 50x_2^2$. For this example it already takes 121 iterations to satisfy the same tolerance level for the gradient of $\hat{f}$.

Newton's method

Newton's method is a line search method, which is actually designed to numerically find roots of equations. But it can be applied to the first-order optimality condition (2.4) to find stationary points of a function. It attempts to find a sequence of points, starting at an initial guess x_0 converging towards a x^* for which $\nabla f(x^*) = 0$ holds. To obtain the search direction in every step a perturbation p_k is added to the current iterate x_k. The perturbed gradient function is then linearized by a first-order Taylor-approximation

$$\nabla f(x_k + p_k) \approx \nabla f(x_k) + \nabla^2 f(x_k) p_k$$

This approximation of the gradient is then set equal to zero, which leads to a set of linear equations

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k).$$

The solution of this system reveals the search direction

$$p_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k).$$

A problem of this method is, that it is equally attracted by all points where the gradient is zero, which can be minima, maxima and saddle points. So it is necessary that the function is locally convex (that means the Hessian matrix has to be positive definite) in order to guarantee that the computed direction is a descent direction.

A way to find out wether the search direction is a descent direction is to check the following condition:

$$\begin{aligned} 0 > f(x_k + p_k) - f(x_k) &\approx f(x_k) + p_k^T \nabla f(x_k) - f(x_k) \\ &= p_k^T \nabla f(x_k) \end{aligned}$$

If p_k does not lead to decreasing values of f it might be advisable to use a different method to compute a search direction (e.g. steepest-descent), or to choose a different initial guess.

Another problem occurs, if the hessian matrix $\nabla^2 f(x_k)$ is singular. Then the system of equations can't be solved and the method fails.

The example for the steepest descent method

$$\min_x f(x) = x_1^2 + 5x_2^2$$

is quadratic and so the Newton's method would solve it in just one step, because Newton's method computes the root of a quadratic approximation of the function, which in this case is exactly the objective function. This shows that this method can be much more efficient than the steepest descent method.

2.2 Equality-constrained optimization

The problem of equality-constrained optimization consists of the objective function f, which has to be minimized, and a set of constraint functions c_i that have to equal zero.

$$\min_x \ f(x) \quad \text{s.t. } c_i(x) = 0, \quad i \in \mathcal{E}. \tag{2.6}$$

Here again there are numerous approaches to solve this problem. Common methods reformulate the problem to incorporate the objective function and the constraints into a new objective function. So the constrained problem becomes similar to an unconstrained problem, which can be solved by the corresponding but slightly modified methods.

Optimality conditions

Let's say we are at a point x_k where the constraints are satisfied. In order to retain feasibility a search direction p_k from this point has to satisfy

$$0 = c(x_k + p_k)$$

which can be approximated by

$$\approx c(x_k) + \nabla c(x_k)^T p_k = \nabla c(x_k)^T p_k.$$

This means the new point $x_{k+1} = x_k + p_k$ still satisfies the constraints. To assure that p_k produces a decrease in f

$$0 > f(x_k + p_k) - f(x_k) \approx \nabla f(x_k)^T p_k$$

has to be satisfied. If there exists a p_k that satisfies both conditions, we can assume, that an improvement on the current iterate x_k is possible. Obviously these conditions can not be satisfied if $\nabla f(x_k)$ and $\nabla c(x_k)$ are parallel, which can be expressed by the condition

$$\nabla f(x_k) = \lambda_k \nabla c(x_k).$$

If there exists no p_k for which these conditions are fullfilled, one could expect x_k to be a minimum. But a simple example shows that this is not the case.

Example 2.2.

$$\begin{aligned} f(x) &= x_1 + x_2 \\ \text{s.t. } c(x) &= x_1^2 + x_2^2 - 2 \end{aligned}$$

Derivatives:

$$\begin{aligned} \nabla f(x) &= (1,1)^T \\ \nabla c(x) &= (2x_1, 2x_2)^T \end{aligned}$$

The point $(1,1)^T$ satiesfies the condition $\nabla f(x) = \lambda \nabla c(x)$ with $\lambda = \frac{1}{2}$. But this point is obviously a maximum, so this condition is *necessary*, but not *sufficient* to characterize a minimum.

These two conditions lead to the introduction of the *Lagrange-multipliers* λ and the *Lagrange function* defined by

$$\mathcal{L}(x, \lambda) = f(x) - \lambda c(x).$$

The first-order conditions can be summarized by

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla_x f(x^*) - \lambda^* \nabla_x c(x^*) = 0 \tag{2.7}$$

Quadratic penalty method

The quadratic penalty method formulates a new objective function adding a penalty term to the original objective function. The penalty term penalizes the new objective function by the square of the infeasibilities. As the constrainets get squared the new objective function retains the properties of smoothness and differentiability. An additional variable is introduced to weight the penalty term. So the quadratic penalty function has the following form:

$$Q(x;\mu) = f(x) + \frac{\mu}{2}\sum_{i\in\mathcal{E}} c_i^2(x) \tag{2.8}$$

For increasing μ it becomes clear, that the violation of the constraints will decrease.

To use the quadratic penalty function as the objective function in Newton's method the first- and second-order derivatives are needed.

$$\nabla_x Q(x;\mu) = \nabla f(x) + \mu\sum_{i\in\mathcal{E}} c_i(x)\nabla c_i(x)$$
$$\nabla_{xx} Q(x;\mu) = \nabla^2 f(x) + \mu\sum_{i\in\mathcal{E}} c_i(x)\nabla^2 c_i(x) + \nabla c_i(x)\nabla c_i(x)^T$$

Lagrangian method

The Lagrangian method is based on the first-order condition (see 2.7). Its goal is to find a stationary point of the Lagrange function.

$$\nabla_x \mathcal{L}(x,\lambda) = 0$$

To find a root in this function Newton's method can be applied. To do so, the derivatives have to be calculated.

$$\nabla_x \mathcal{L}(x,\lambda) = \nabla_x f(x) - \sum_{i\in\mathcal{E}} \lambda_i \nabla_x c_i(x)$$
$$\nabla_{xx} \mathcal{L}(x,\lambda) = \nabla_{xx} f(x) - \sum_{i\in\mathcal{E}} \lambda_i \nabla_{xx} c_i(x)$$

To be able to iteratively adapt the Lagrange multipliers, it is possible to introduce a vector $z = (x,\lambda)^T$ and then differentiate the Lagrange function with respect to z. Finding a root of this gradient will lead to a stepwise adaption of λ.

Augmented Lagrangian method

The augmented Lagrangian method combines the use of Lagrange multipliers and a quadratic penalty term. So the equality-constrained problem (2.6) can be rearranged

to

$$\mathcal{L}_A(x, \lambda, \mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x).$$

The gradient of the augmented Lagrangian has to be set to zero, in order to find a stationary point.

$$\begin{aligned} \nabla_x \mathcal{L}_A(x, \lambda, \mu) &= \nabla_x f(x) - \sum_{i \in \mathcal{E}} \lambda_i \nabla_x c_i(x) + \mu \sum_{i \in \mathcal{E}} c_i(x) \nabla_x c_i(x) \\ &= \nabla_x f(x) - \sum_{i \in \mathcal{E}} \left(\lambda_i - \mu c_i(x)\right) \nabla_x c_i(x) \end{aligned}$$

An adaption of λ at step k for the i-th component of λ is provided by

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_{k+1})$$

Example 2.3. This example consists of an objective function, which is an inclined plane and a constraint, which forces the solution to be on a circle around $(0, 0)^T$ with a radius of $\sqrt{2}$.

$$\begin{aligned} \min_x f(x) &= x_1 + x_2 \\ \text{s.t. } c(x) &= x_1^2 + x_2^2 - 2 = 0 \end{aligned}$$

The solution is obviously at $(-1, -1)^T$ but there is another point $(1, 1)^T$, where the augmented Lagrangian function is zero. But this point is a maximum. A method to not get attracted by this maximum is to use a merit function, which prevents the algorithm from going into a direction, where the objective function values are increasing.

Figure 2.2 shows the iterations of the algorithm starting at $x_0 = (2, 1.5)^T$ and terminating after 15 iterations at approximately $x^* = (-1, -1)^T$.

2.3 Inequality-constrained optimization

The general optimization problem containing an arbitrary amount of equality and inequality constraints is defined by

$$\min_x \ f(x) \quad \text{s.t. } c_i(x) = 0, \quad i \in \mathcal{E}, \quad c_i \geq 0, \quad i \in \mathcal{I}.$$

Adaption of the augmented Lagrangian method

The idea of the adaption of the augmented Lagrangian method to inequality constraints is, that a selection of inequality constraints is treated as equality constraints.

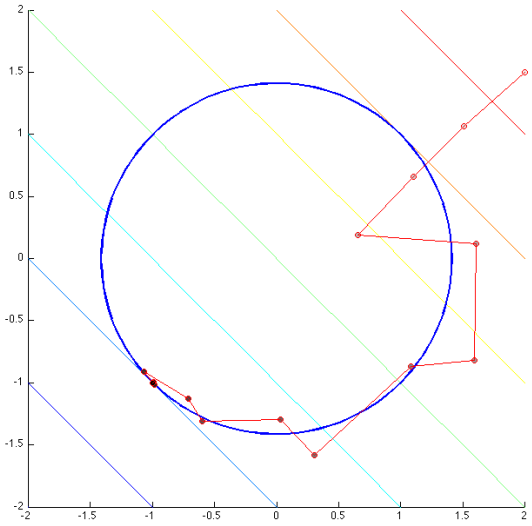


Figure 2.2: Example of an equality constrained optimization with the augmented Lagrangian method

This selection is called *working set*. There are certain rules handling when a constraint is added to, or removed from, the working set. Of course, if there are any equality constraint, they always have to be in the working set. When the algorithm starts all inequality constraints for which $c_i(x) < 0$ holds are added to the initial working set. In the phase of the step length computation it is checked which constraints are restricting a movement in the search direction. The last restricting constraint is then added to the working set. And finally at the preparation of the next step the constraints whose Lagrange multipliers would become negative are removed from the working set. To explain why inequality constraints with negative Lagrange multipliers can be neglected for the next iteration step, I will give a small example.

Example 2.4. The example uses the same functions as in the example for equality constraints, with the difference that the constraint is multiplied by -1. This means that the solution has to be inside or on the circle.

$$\begin{aligned} \min_x f(x) &= x_1 + x_2 \\ \text{s.t. } c(x) &= 2 - x_1^2 - x_2^2 - 2 \geq 0 \end{aligned}$$

The gradient of the constraint is always pointing towards the inside of the feasible region. As the method is trying to find a point where the gradient of the constraint and the objective function are parallel, there are two possiblities: either they point in the same direction (in this case λ is negative) or they point in the opposite direction

(λ positive). If a point is reached where they point in the same direction, there exists a direction where the objective function is decreasing (negative gradient) and where the inequality constraint holds (gradient of the constraint function). So in this case it makes no more sense to stay on the constraint border, thus the inequality constraint must not be treated as an equality constraint but has to be neglected.

Example 2.5. This example shows how the working set influences the progress of the algorithm. The solution of this problem is at $x^* = (-\sqrt{2}, 0)^T$.

$$\begin{aligned} \min_x \; & f(x) = (x_1 + 2)^2 + (x_2 + 2)^2 \\ \text{s.t. } & c_1(x) = 2 - x_1^2 - x_2^2 - 2 \geq 0 \\ & c_2(x) = x_2 \geq 0 \end{aligned}$$

Figure 2.3 shows the iterations of the algorithm starting at the feasible point $x_0 = (0.5, 1)^T$ and s in the first iteration no constraint is active it simply tries to minimize the unconstrained objective function, but the constraint c_2 is restricting the search direction, so for the next iteration is added to the working set. The next step shows how the search direction is moving along the constraint border of c_2. But then the constraint c_1 is the restricting constraint so the next working set consists of c_1 and again one can see how the search direction aligns to this constraint. The algorithm moves on like this until it has reached a satisfiable result.

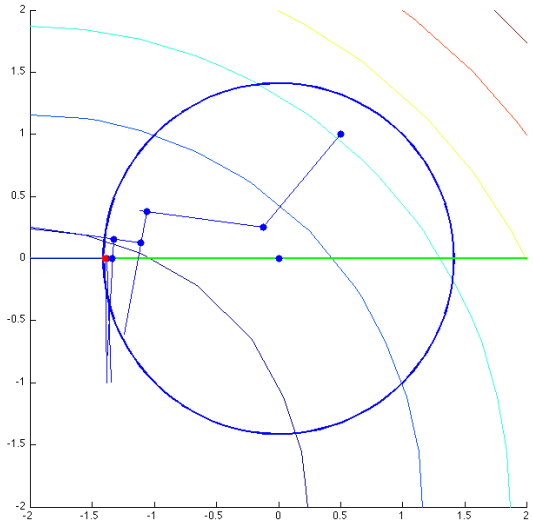


Figure 2.3: Example of an inequality constrained optimization with the augmented Lagrangian method

3 Implementation

The goal of this chapter is to give an understanding of how I realized some of the algorithms explained in the previous section with MATLAB.

First this chapter shows common properties of the implementations of the algorithms, second I will have a closer look at the most important algorithms and explain them in detail. I first want to present how the function interfaces work as this is common to all functions. Basically a MATLAB routine who optimizes a function only needs two input arguments: the target functions, consisting of the objective function and constraint functions for constrained optimization, and an initial guess. However there are several settings that can be of interest, depending on the problem that has to be solved. So there is a set of optional parameters, that can be adapted.

```
[x] = function_name(objFctn, constrFctn, xc, varargin)
```

Listing 3.1: Interface of a general optimization method

The function handles have to be implemented in a separate MATLAB-function and have the following interface:

```
[f, df, d2f] = function_name(x)
```

Listing 3.2: Interface of an objective function

The output arguments correspond to $f(x)$, $\nabla_x f(x)$ and $\nabla_{xx} f(x)$.

The constraint functions are handled similarly. A difference exists when there are multiple constraints. If there are m constraints the constraint function handle represents $c : \mathbb{R}^n \to \mathbb{R}^m$. So `f` is a column vector with m entries, `df` represents the Jacobian-matrix of c and `d2f` is a 3-dimensional matrix $\in \mathbb{R}^{n \times n \times m}$ consisting of m Hessian-matrices.

The general structure of the programs follows the steps

1. Initialize variables, set standard/custom settings
2. Start iteration
    a) Compute function values
    b) Check break-conditions

c) Compute search direction

d) Compute step length

e) Set values for the next iterate

3. End iteration, if program ends here, the algorithm failed to find a minimizer

3.1 Newton's method

The first version of a Newton's method algorithm

```
[x] = Newton(objFctn, xc, varargin)
```

Listing 3.3: Interface of the Newton's method

As Newton's method solves unconstraind optimization problems the input only consists of an objective function, an initial guess and a list of optional arguments.

Parameter	Explanation
maxIter	Maximum number of over all iterations
LSMaxIter	Maximum number of iterations for the line search
tolF	Tolerance value for the change of the function values
tolX	Tolerance value for the change in x
tolG	Tolerance value for the gradient
theta	Coefficient for the merit function (Armijo condition)
doPlot	Parameter for plotting the optimization process

Table 3.1: List of optionally modifiable parameters for Newton's method

The algorithm starts by initializing some parameters by its standard values, after they are initialized the parameters are overwritten by custom values, if they are set in the function call. The next step is to compute the function values at the initial guess, set some values for the x and the function values at the last step, which are needed for the break-conditions, and finally setting the iteration counter to zero.

Break conditions

Then the iteration starts by checking the break-conditions. There are four conditions:

1. Change in f: This condition checks if the change of the function value compared to the last step is below a certain tolerance level.

2. Change in x: Here the norm of the difference of x_k and x_{k-1} is compared with a tolerance value.

3. Norm of the gradient: Check if the norm of the gradient of the objective function is below the tolerance.

4. Number of iterations: Check if the maximum of iteration steps has been reached.

In the first iteration the condition 1 and 2 are set to `false`, because the last step information is not yet meaningful. The algorithm terminates if either the conditions 1 and 2 are or at least one of the conditions 3 and 4 are satisfied. The sense of these conditions is that if the first two are satisfied there is not much improvement of the function to be expected in further iterations. Condition 3 is the condition that hopefully is reached, because it means that a stationary point has been found, which should be a minimum. Condition 4 is fullfilled if no stationary point has been reached after the maximum of iterations.

Search direction

If the algorithm did not terminate due to the break-conditions it continues by computing a search direction. For this purpose the gradient and the Hessian matrix of the objective function are needed. Before the search direction is obtained with Newton's method, the algorithm checks if the conditioning of the Hessian matrix is good enough for MATLAB's backslash-operator to solve the linear equation system. If this is not the case, the search direction is set to the negative gradient i.e. the steepest descent direction.

Step length

The next phase of the algorithm is computing a step length γ. First the merit function is defined by

$$m(\gamma) = f(x_k) + \gamma\theta\nabla f(x_k)^T p_k$$

The initial step length is set to 1. Then an iteration starts which breaks if the gradient has decreased and the *Armijo condition* is satisfied. As long as this is not the case each iteration reduces the step length by multiplying it with the factor $\frac{1}{2}$.

3.2 Augmented Lagrangian for inequality constraints

The variant of the augmented Lagrangian method presented here will be used for the application to the SPECT reconstruction, so it is specialized for inequality constraints only. It handles only a single nonlinear inequality constraint and an arbitrary number of linear bound constraints.

```
[x] = AugmentedLagrange(fctn, constr, boundconstr, xc,
   varargin)
```

Listing 3.4: Interface of the augmented Lagrangian method

Parameter	Explanation
pc	Lagrange multipliers
mu	Initial weight of the penalty term
maxIter	Maximum number of over all iterations
LSMaxIter	Maximum number of iterations for the line search
tolF	Tolerance value for the change of the function values
tolX	Tolerance value for the change in x
tolG	Tolerance value for the gradient
theta	Coefficient for the merit function (Armijo condition)
doPlot	Parameter for plotting the optimization process

Table 3.2: List of optionally modifiable parameters for the augmented Lagrangian method

Working set

An important part of this algorithm is the concept of a working sets containing the active constraints. The active constraints are limiting the search direction to move towards decreasing objective function values. In order to retain feasibility these active constraints of the working set have to be considered for the computation of the search direction. The constraints not in the working set can be neglected for the search direction they are regarded when it comes to defining the step length.

The working set in the program is represented by a logical variable, which is responsible for the nonlinear inequality constraint and an array of logical variables representing the active bound constraints. The vector λ contains only the Lagrange multipliers of the active constraints. To complete the augmented Lagrangian function only the constraint function values of the active constraints are considered. As the bound constraints are linear in this case the second derivative of the constraints consists only of the second derivative of the non-bound constraint.

The definition of the new working set is done partly during the step length computation. There the last constraint which was inflicted gets added to the working set of the next iteration. After the step length computation the algorithm checks if there are any constraints that can be removed from the working set. This is the case for constraints where the corresponding Lagrange multiplier are negative.

4 Application: Single Photon Emission Computed Tomography

In this part I want to apply the implemented algorithms to a real problem: the reconstruction of data from *Single Photon Emission Computed Tomography*. First I will outline how SPECT works based on [2] and how a SPECT measurement can be simulated in MATLAB. Then I will propose some mathematical models, which are the basis for a reconstruction with optimization and finally I will evaluate the results of my reconstruction algorithms concerning the different models, trying to show that a more sophisticated model including constraints achieves better results.

4.1 SPECT functionality

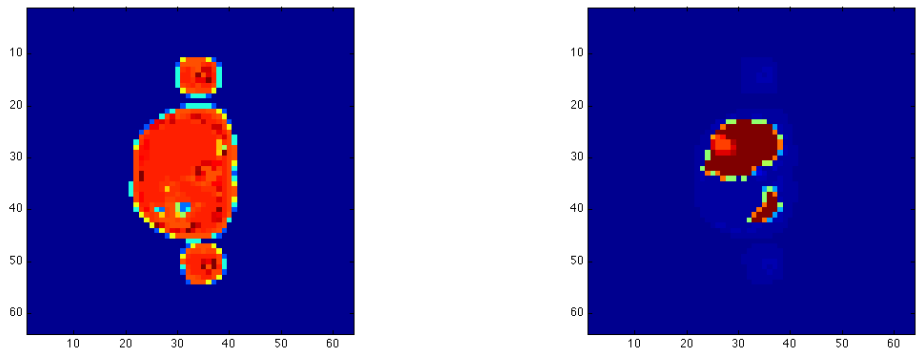
Single photon emission computed tomography is a medical imaging method, which allows to visualize metabolic functions of a patient. Other than in x-ray tomography, where the attenuation of rays going through an object is measured, the radiation emitting source in SPECT is situated inside the object. The radiation is emitted by radioactive isotopes placed inside a patient by e.g. injection of a radiopharmaceutical. The goal is to obtain the concentration of this radiopharmaceutical inside the body, which is proportional to the amount of gamma-ray photons emitted.

The measuring device consists of an array of sensors, which count the number of gamma-ray photons emitted on a line perpendicular to the sensor device. To ensure that only photons on this specific line are respected, a collimator is placed in front of the detector absorbing the radition, which is not perpendicular to the device. In reality this can not be accomplished completely. If such a projection is recorded the device is rotated around the patient to the next angle position to record the next projection. It is obvious that the 3-dimensional distribution of the pharmaceutical has to be obtained, but for the sake of simplicity and as proposed by other authors [1] the problem can be treated as a set of 2D problems, considering the object as a set of 2-dimensional cross sections.

As the radiation source is surrounded by tissue, the radiation is attenuated on its way to the detector. This is a fact that makes SPECT much harder than for example CT, as the problem here consists of two unknown images: the activity distribution f and the attenuation map μ, which is normally not known. For the calculations performed in this thesis, I will however take μ for granted.

4.2 SPECT simulation

To simulate a SPECT measurement I was using a phantom (by courtesy of Sven) consisting of two 3-dimensional models of a complete human body representing the attenuation (i.e. the tissue density) and the activity (i.e. the concentration of a marker) distribution.



(a) (b)

Figure 4.1: A slice of the phantom with the attenuation map (a) and the activity distribution (b)

Figure 4.1 shows a 2-dimensional slice of the phantom. The original resolution of a phantom slice is 512x512, but in order to reduce the computation time and the memory space needed for both simulation and reconstruction I reduced the resolution to 64x64 using the routine `linearInter2D` from the FAIR framework.

In order to simulate a SPECT measurement based on these phantom data, I used two different MATLAB programs. One was implemented by me and the other one was implemented by Sven Barendt. I will roughly describe how my implementation works, but for the generation of the testdata I used Sven's program, because it is more sophisticated and thus provides much better results. Both algorithms allow to extract the backprojection matrix, which is necessary for my reconstruction algorithm.

The input arguments for the routine consist of the attenuation map `Bmu`, the activity distribution `Bf`, the number of collimators on the sensor device `nP` and the number of angles the sensor device is recording data `nPhi`. The function returns the Sinogram recorded by the device `g`, where the each column represents the recorded data from one angle and the projection matrix `P`, where each row contains the attenuation factor of every pixel on the line from a collimator at a specific angle.

At first the coordinates of the lines pointing from each collimator at each angle are computed. Then the algorithm determines which pixels of the phantom image are on these lines and calculates for each such pixel the factor by which the radiation from this point gets attenuated on their way to the detector. This factor b can be calculated by

$$b = \alpha e^{-\alpha \sum_{i \in \mathcal{T}} \mu_i}.$$

In this formula the set $\mathcal{T}$ contains the indices of the pixels that are on the line from the pixel to the recording device.

After having calculated the projection matrix P, consisting of all the attenuation factors, the sinogram can be simply computed by a matrix multiplication $g = P \cdot f$. It provides the sinogram as a column vector, which can be reshaped, so that each column pictures the data measured by the collimator array at a specific angle. Figure 4.2 and 4.3 provide examples for such sinograms, computed by the two different algorithms. They both were simulated with 64 collimators and 64 angle positions.

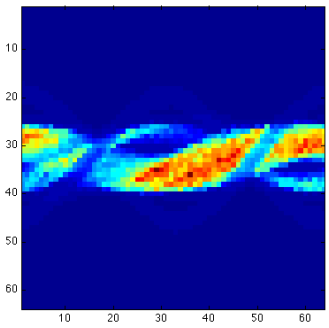


Figure 4.2: Sinogram of my simulation algorithm

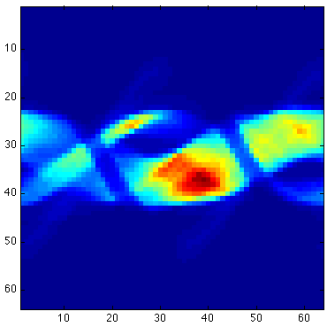


Figure 4.3: Sinogram of Sven's simulation algorithm

4.3 Reconstruction strategies

In order to reconstruct the SPECT data with optimization, a mathematical model must be chosen. I will propose some models, that I will also test later.

The most basic model is to minimize the quadratic error function, defined by

$$\min_f \mathcal{J}(f) = \frac{1}{2} \|P_\mu f - g\|_2^2$$

without constraints. But it is quite obvious that this model also allows certain elements of f to be negative, which does not consider the physical meaning of f. As f is actually

measuring the intensity of radiation reaching the device, a negative intensity is never feasible. So a more advanced model could demand nonnegativity for f:

$$\begin{aligned} \min_f \mathcal{J}(f) &= \frac{1}{2}\|P_\mu f - g\|_2^2 \\ \text{s.t. } c(f) &= f \geq 0 \end{aligned}$$

The constraint here is actually a set of constraints with $c_i(f) = f_i \geq 0$ for $i = 1 \dots N$, where N is the number of pixels.

Another strategy is to accept a certain error A in the quadratic error function (maybe assuming a non perfect sinogram or projection matrix) and to focus more on minimizing the total activity. This leads to the following problem statement

$$\begin{aligned} \min_f \mathcal{K}(f) =& \|f\|_2^2 \\ \text{s.t. } \mathcal{J}(f) &= A - \frac{1}{2}\|P_\mu f - g\|_2^2 \geq 0 \\ c(f) &= f \geq 0 \end{aligned}$$

4.4 Evaluation

In this section I will compare different reconstruction strategies all starting from an initial guess where the activity on every pixel is zero.

Setting 1

The first setting is a simple quadratic error minimization.

$$\min_f \mathcal{J}(f) = ||P_\mu f - g||_2^2$$

The initial guess is a matrix filled with zeros and the the optimization method used is actually Newton's method, but in practice the Hessian matrix of the objective function was always ill-conditioned, so the algorithm automatically chooses the steepest descent direction instead. The reason why the Hessian was ill-conditioned simply comes from the fact that the simulation gathered not enough information to solve the system directly. If Newton's method would work, the solution could be computed in one iteration.

Setting 2

The second setting minimizes the quadratic error function but the solution is restricted to nonnegative activity.

$$\begin{aligned} \min_f \mathcal{J}(f) &= \frac{1}{2}\|P_\mu f - g\|_2^2 \\ \text{s.t. } c(f) &= f \geq 0 \end{aligned}$$

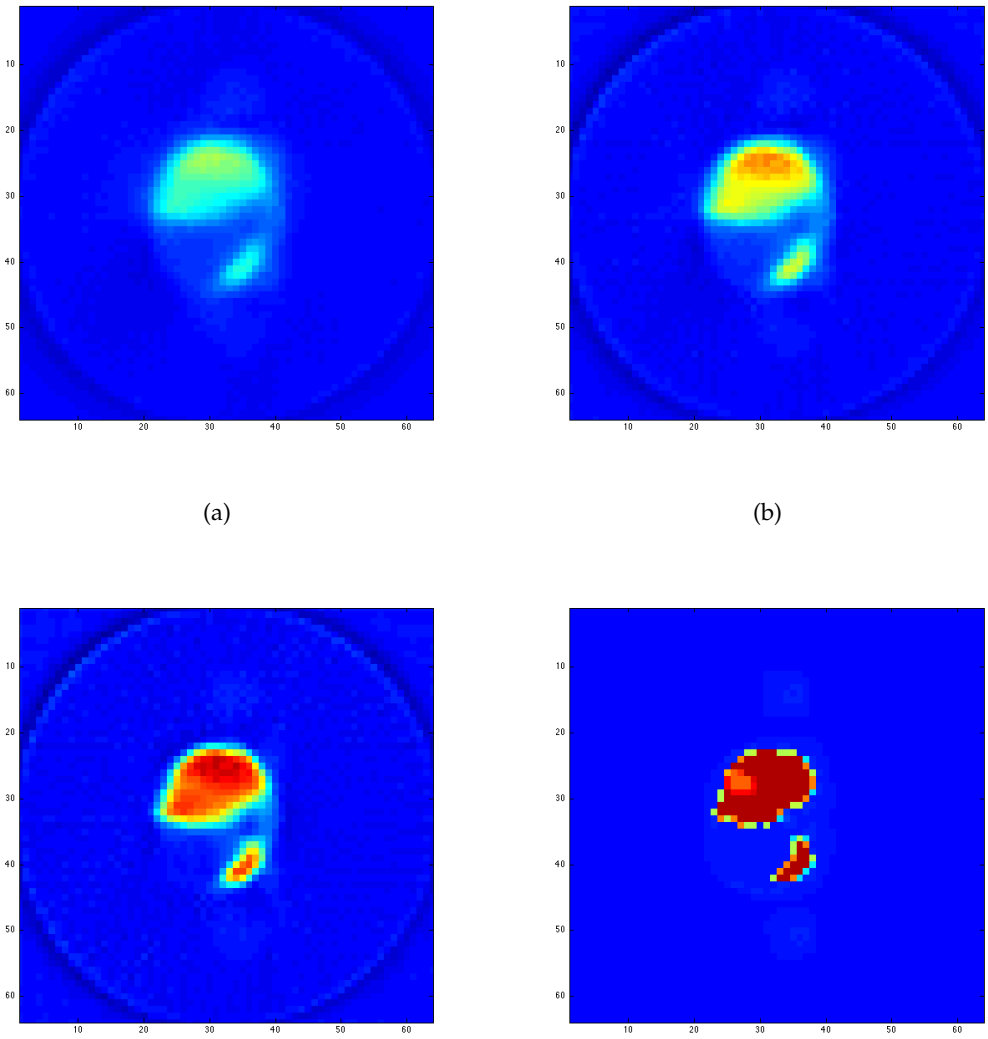
Setting 1 and 2 both are very similar, they both show the circular artifact and converge in about the same rate towards the original image. Although setting 2 is slightly faster.

Setting 3

The third setting is attempting to minimize the overall activity, restricted by some quadratic error constant and nonnegative activity.

$$\begin{aligned} \min_f \mathcal{K}(f) =& \|f\|_2^2 \\ \text{s.t. } \mathcal{J}(f) &= A - \frac{1}{2}\|P_\mu f - g\|_2^2 \geq 0 \\ c(f) &= f \geq 0 \end{aligned}$$

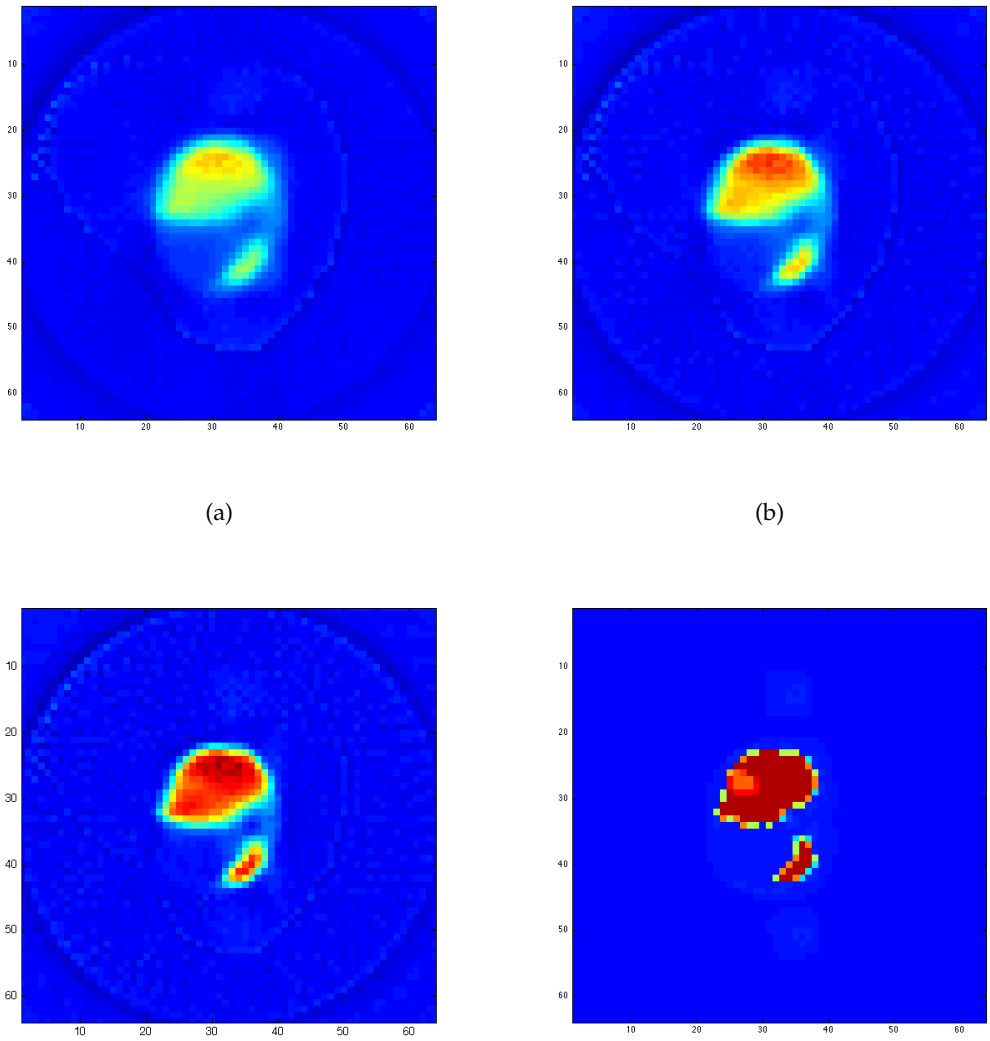
I have tried out some different values for A, but the difference in the result was vanishingly small. The difference compared to setting 1 and 2 is remarkable. Especially the low number of iterations needed to reach the activity level of the original concerning the critical parts of the image. Also the edges of those parts are very clearly present from the first iterate on.



(a) (b)

(c) (d)

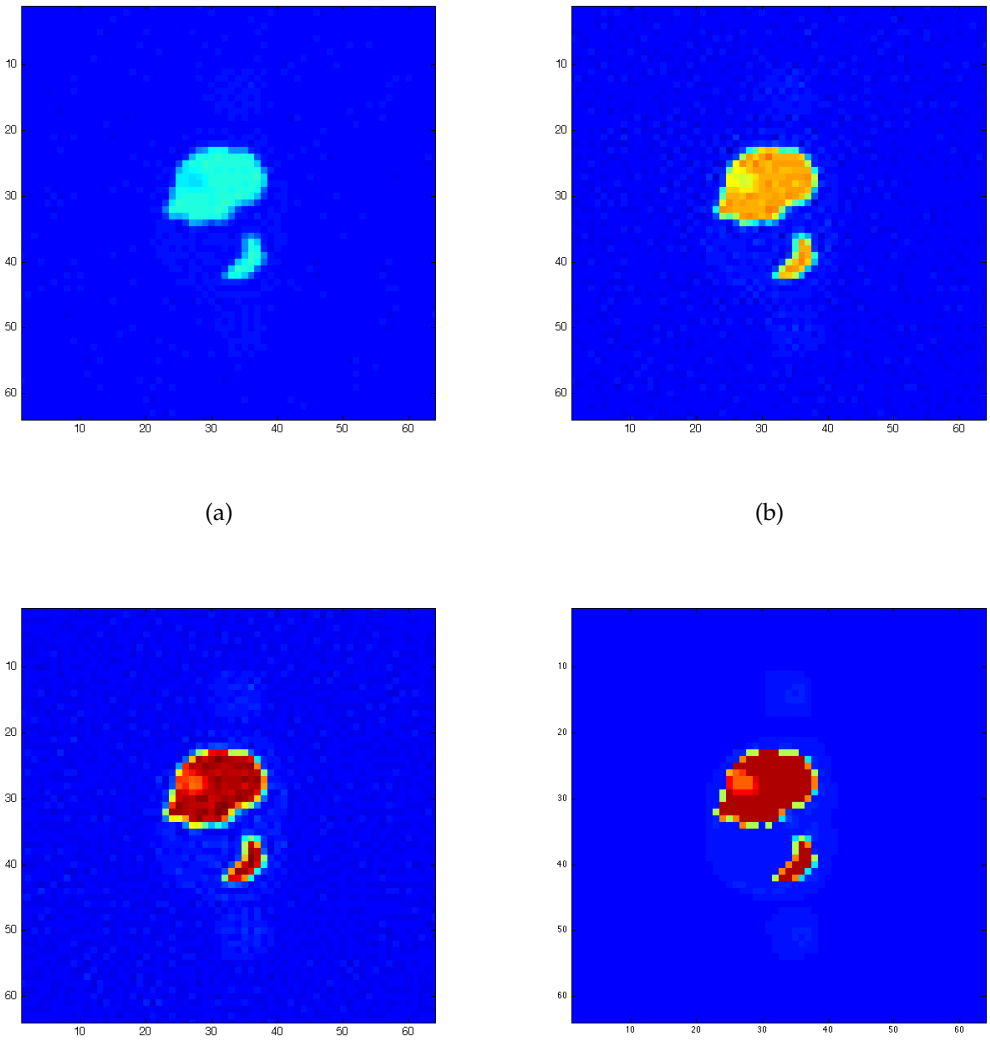
Figure 4.4: Reconstruction of the phantom slice with setting 1 depicted after 10 (a), 20 (b) and 50 (c) iterations compared to the original phantom data (d).



(a) (b)

(c) (d)

Figure 4.5: Reconstruction of the phantom slice with setting 2 depicted after 10 (a), 20 (b) and 50 (c) iterations compared to the original phantom data (d).



(a) (b)

(c) (d)

Figure 4.6: Reconstruction of the phantom slice with setting 3 depicted after 1 (a), 3 (b) and 10 (c) iterations compared to the original phantom data (d).

5 Conclusion

To conclude the thesis I want to summarize what has been achieved and propose what could be done to improve the implementation for practical applications.

As shown in the former section the introduction of constraints for the reconstruction of SPECT data leads to a significant improvement in the convergence rate of the algorithm and also on the fidelity of the reconstructed image. But the algorithm could be further improved by replacing the MATLAB's backslash operation by a conjugate gradient technique. This would make the algorithm more robust. Another improvement could be achieved in the evaluation process by trying to create a more realistc SPECT simulation. This could be achieved by simulating scattering effects during the measuring process, they result in a random noise on the sinogram. I expect that for this setting the third reconstruction strategy is even better than with the settings I tested.

Bibliography

[1] V. Dicken. *Tikhonov-IntraSPECT*. PhD thesis, University Potsdam, 1997.

[2] A. C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[3] J. Modersitzki. *FAIR - Flexible Algorithms for Image Registration*. Oxford Press, 2000.

[4] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

List of Figures

List of Tables

Listings