Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Matematica

REGULARIZATION METHODS FOR THE SOLUTION OF A NONLINEAR LEAST-SQUARES PROBLEM IN TOMOGRAPHY

Tesi di Laurea in Analisi Numerica

Relatore:
Chiar.ma Prof.ssa
ELENA LOLI PICCOLOMINI

Correlatore:
Chiar.ma Prof.ssa
GERMANA LANDI

Presentata da:
STEFANO BERNARDINI

I Sessione

Anno Accademico 2014/15

Ai miei nonni e allo zio Cecco

Introduction

The need to reconstruct images from signals arises in several fields and applications. Among these, one of the most relevant is the *Medical imaging*, that is the sector responsible for producing some reconstructed images of the human body for a diagnostic and therapeutic aim. One of the most current diagnostic imaging technique is the Computerized Tomography (CT), namely a methodology that allows to reproduce sections of patient's body via the emission of ionized radiation beams (X-rays) and permits a three-dimensional reconstruction thanks to the aid of a calculator. More recently, the new technique of Digital Tomosynthesis (DT) has got increasing interest. DT is a methodology for reconstructing a three-dimensional object by using only a finite number of projections over a limited range of angles. For this reason, it is particularly suitable for the analysis of fragile parts of the body because they are subject to a smaller quantity of radiations.

Usually in the most common DT systems the mathematical model approximates the reality by considering the object made of only one material and the X-rays beam made of only one energy level. The contribution of this thesis is to study a polyenergetic and multimaterial model for a breast image reconstruction in Digital Tomosynthesis, namely a more complex mathematical model that takes into account the variety of the materials forming the object and the polyenergetic nature of the X-rays beam. The modelling of

an object composed of several materials permits to separate the different tissues composing the object, allowing a more informative reconstruction. On the other hand, considering a polyenergetic radiation beam permits to avoid the appearance of artifacts linked to the monoenergetic nature of X-rays and then to get a higher quality reconstructed image, making sometimes a post-processing phase unnecessary.

Unfortunately the image reconstruction represents the resolution of an inverse ill-posed problem, that needs a regularization. Moreover, that model leads to the resolution of a high-dimensional nonlinear least-squares problem, that is much more complex than a linear least-squares one and needs for its resolution a relevant quantity of storage and time. For these reasons, we test several numerical methods fit to solve this type of problem; among them, the principal ones are the Levenberg-Marquardt method with different possible choices for the regularization parameter and two algorithms belonging to the L-BFGS class, i.e. those methods that implement the BFGS formulas but need only a limited quantity of storage.

In the first chapter of this thesis the fundamental concepts of the unconstrained regularization are presented and, after that the two different strategies of line search and trust region have been introduced, the several methods for the resolution of a nonlinear least-squares problem that will be used in the experiments are analysed.

In the second chapter, after having presented briefly the Computerized Tomography and Digital Tomosinthesys techniques, we turn to the multi-material and polyenergitic model. For this goal we create a test problem composed by an object with a known materials composition in order to try to reconstruct it via the numerical methods presented.

In the third chapter we make the real experiments of the introduced nu-

merical methods. Since the tests at the calculator take much time, firstly we analyse the results of the various methods for a test problem of limited dimensions. Later, once identified the methods and parameters corresponding to the best solutions, we test them for a higher-dimensional problem, comparing the reconstructed images qualities too.

In the Conclusions we make a synthesis of the results we got and identify those methods that, in our experiments, turn out to be the most efficient ones.

Introduzione

La necessità di ricostruire immagini a partire da segnali nasce in diversi campi ed applicazioni. Fra questi, uno dei più rilevanti è il *Medical imaging*, ovvero il settore che si occupa di fornire delle ricostruzioni di immagini del corpo umano per fini diagnostici e terapeutici. Una fra le tecniche di diagnostica per immagini più diffuse è la Tomografia Computerizzata (CT), ovvero una metodologia che permette la riproduzione di sezioni del corpo del paziente per mezzo dell'emissione di fasci di radiazioni ionizzate (raggi X) e ne permette una ricorstruzione tridimensionale grazie all'ausilio di un calcolatore. Più recentemente sta attirando sempre più interesse la tecnica di Tomosintesi Digitale (DT). La DT è una metodologia atta a ricostruire un oggetto tridimensionale utilizzando solo un numero finito di proiezioni su un range limitato di angolazioni. Per questi motivi, essa risulta particolarmente indicata per l'analisi di parti delicate del corpo perchè soggette ad una quantità minore di radiazioni.

Solitamente nei più diffusi sistemi DT il modello matematico approssima la realtà considerando l'oggetto composto da solo un materiale e il fascio di raggi X ad un solo livello di energia. Il contributo di questa tesi è quello di studiare un modello polienergetico e multimateriale per la ricostruzione dell'immagine di una mammella attraverso la tecnica di Tomosintesi Digitale, ovvero un modello matematico più complesso che tenga in considerazione

la varietà dei materiali di composizione dell'oggetto e la natura polienergentica del fascio di radiazione X. La modellizzazione di un oggetto composto da diversi materiali consente di separare i diversi tessuti che compondono il corpo, permettendo una ricostruzione più ricca di informazioni. Dall'altro lato considerare la radiazione a più livelli di energia permette di evitare la comparsa di artifatti legati alla natura monoenergetica dei raggi X e quindi di ottenere un'immagine ricostruita di qualità superiore, rendendo a volte superflua una fase di *post-processing*.

Sfortunatamente la ricostruzione dell'immagine costituisce la risoluzione di un problema inverso mal posto, che necessita quindi di una regolarizzazione. Inoltre, tale modello conduce alla risoluzione di un problema di minimi quadrati non lineari di grandi dimensioni, che risulta assai più complesso di un problema di minimi quadrati lineari e che necessita per la sua risoluzione una consistente quantità di memoria e di tempo. Per questi motivi sono stati testati diversi metodi numerici adatti alla risoluzione di un simile problema; tra questi, i principali sono il metodo di Levenberg-Marquardt con diverse possibili scelte del parametro di regolarizzazione e due algoritmi della classe L-BFGS, ovvero quei metodi che implementano le formule BFGS ma che necessitano solo di una quantità limitata di memoria.

Nel primo capitolo di questa tesi sono stati introdotti i concetti fondamentali dell'ottimizzazione non vincolata e, dopo aver presentato le due diverse strategie di ricerca in linea e *trust region*, si sono analizzati i vari metodi per la risoluzione di un problema di minimi quadrati non lineari che saranno utilizzati nelle sperimentazioni.

Nel secondo capitolo, dopo aver presentato brevemente la tecnica della Tomografia Computerizzata e della Tomosintesi Digitale, si è passati al modello multimateriale e polienergetico. Per far ciò si è costruito un problema

test costituito da un oggetto dalla composizione di materiali nota con l'obiettivo di tentare di ricostruirlo attraverso i metodi numerici presentati.

Nel terzo capitolo si è proceduti alla vera e propria sperimentazione dei metodi numerici introdotti. Siccome le prove al calcolatore richiedono molto tempo, si è dapprima analizzato i risultati dei vari metodi per un problema test di dimensioni limitate. Dopodiché, individuati i metodi e i parametri che restituiscono le soluzioni migliori, si è proceduto a testarli su un problema di dimensioni maggiori, confrontando anche la qualità delle immagini ricostruite.

Nelle Conclusioni faremo una sintesi dei risultati ottenuti e individueremo i metodi che, nei nostri test, si sono rivelati i più efficienti.

Contents

List of Figures

List of Tables

Chapter 1

Some unconstrained optimization methods

In this chapter we are going to show the fundamentals of unconstrained optimization and give all the instruments that we need to solve our problem of image reconstruction in tomography. So, first we will make an overview of a minimization problem and we will present the most important definitions and theorems. After having presented the two main strategies into which all minimization problems are broken down (*line search* and *trust region* methods) and some methods (the Conjugate Gradient method and the CGLS method), we will focus on the problem of uncostrained optimization of a nonlinear least-squares problem and we will show some strategies to solve it. Finally, we will show how the gradient vector can be approximated, since we will need it in the future.

1.1 Fundamentals of unconstrained optimization

The goal of the unconstrained optimization is to minimize a certain function, usually called *objective function*, that depends on real variables, without any constraints or restrictions on these variables. More formally, we intend to find

$$x^* = \arg\min_{x \in \mathbb{R}^n} f(x)$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function, which we suppose at least differentiable.
The best we could find is a *global minimizer* of f, namely a point where the function reaches its least value. Formally:

Definition 1.1.1. *A point x^* is a **global minimizer** if $f(x^*) \leq f(x)$ for all x in the domain of interest to f.*

Since a general algorithm does not visit many points of f, and then we do not have a precise picture of its shape, a global minimizer is often difficult to find, also because we cannot know if the function takes some sharp dips between an evaluated point and the next one. So, most algorithms intend to find a *local* minimizer, namely a point where f reaches the least value relatively to a certain neighbourhood of the point:

Definition 1.1.2. *A point x^* is a **local minimizer** if exists a neighbourhood N of x^* such that $f(x^*) \leq f(x)$ for all $x \in N$.*

From the definition above, it might seem that in order to recognize a minimizer we should examine all the points in the domain of interest of f or in any neighbourhood of the point, and this would be clearly impossible.

However, if the function is smooth, there are some more efficient and practical criteria in order to identify local minimizers. The most important ones come all from the central tool used to study optimization problems, the Taylor's theorem:

Theorem 1.1.1 (Taylor's theorem)**.** *Suppose that* $f \in C^1(\mathbb{R}^n, \mathbb{R})$ *and* $p \in \mathbb{R}^n$*. Than we have that*

$$f(x+p) = f(x) + \nabla(x+tp)^T p$$

for some $t \in (0,1)$*. Moreover, if* $f \in C^2(\mathbb{R}^n, \mathbb{R})$*, we have that*

$$\nabla f(x+p) = f(x) + \int_0^1 \nabla^2 f(x+tp)p \; dt$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp)p \tag{1.1}$$

for some $t \in (0,1)$*.*

From this theorem we can get some necessary and sufficient conditions that will be very useful to check local minima. For example, if $f \in C^2(\mathbb{R}^n)$ we can get some information by examining just the gradient $\nabla f(x^*)$ and the Hessian $\nabla^2 f(x^*)$. Before just enunciating the most important ones, we give a definition.

Definition 1.1.3. *A point* x^* *is called a* ***stationary point*** *if* $\nabla f(x^*) = 0$*.*

Theorem 1.1.2 (First-Order Necessary Conditions)**.** *If* x^* *is a local minimizer and* $f \in C^1$ *in any open neighbourhood of* x^**, then* x^* *is a stationary point.*

Theorem 1.1.3 (Second-Order Necessary Conditions)**.** *If x^* is a local minimizer and $f \in C^2$ in any open neighbourhood of x^*, then x^* is a stationary point and $\nabla^2 f(x^*)$ is positive semidefinite.*

Theorem 1.1.4 (Second-Order Sufficient Conditions)**.** *If $f \in C^2$ in any open neighbourhood of x^*, x^* is a stationary point and $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local minimizer of f.*

Sometimes having some additional information about f can help us to identify global minimizers. As the next theorem states, an important special case is that of convex functions:

Definition 1.1.4. *A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a **convex function** if*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

for all $0 \leq \alpha \leq 1$ and for all $x, y \in \mathbb{R}^n$.

Theorem 1.1.5 (Optimality Conditions for Convex Functions)**.** *If f is a convex function, then every local minimizer is a global minimizer. If f is a strict convex function, then exists only one global minimizer. If f is a differentiable convex function, then every stationary point is a global minimizer.*

1.2 Overview of algorithms

Every algorithm for unconstrained minimization requires the user to give a starting point, which is usually called x_0. If the user has some knowledge about the application and the data set, he may be in a good position to choose the starting point x_0 to be a reasonable estimate of the solution. Otherwise, if the user has no knowledge about it, then the starting point

must be chosen by the algorithm, either by a systematic approach or in some arbitrary way.

The optimization algorithms, starting from x_0, generate a sequence of iterates $\{x_k\}_{k=0}^{\infty}$ that stops when either a solution point has been approximated with sufficient accuracy, either no more progress can be made or when the maximum number of iterations is reached. In order to move from one iterate x_k to the next, the algorithms use some information about f in x_k (can be f itself, or its derivative, and so on) and in some earlier iterates $x_0, x_1, \ldots, x_{k-1}$ with the goal to find a point with a lower objective function value. Every algorithm differs from the others in the way it chooses the next iterate from earlier iterates.

Now we present the two cathegories into which the collection of algorithms for unconstrained optimization of smooth functions can be broken down: *line search* algorithms and *trust region* algorithms.

1.2.1 Line search algorithms

In the line search strategy, the algorithm at every step x_k first chooses a direction p_k and than searches along this direction from x_k a new iterate with a lower objective function value. The distance to move along p_k, namely the step length α_k, can be found by approximately solving the following one-dimensional minimization problem:

$$\arg\min_{\alpha>0} f(x_k + \alpha p_k) \tag{1.2}$$

If we could solve (1.2) exactly, then we would derive the maximum decrease from the direction p_k, but an exact minimization may be expensive and is usually unnecessary. Instead, the line search algorithm generates a limited

number of trial step lengths until it finds one that loosely approximates the minimum of (1.2). When, at the new point, a new search direction and step length are computed, then the process is repeated. So, more formally, a line search method can be written in the form:

$$x_{k+1} = x_k + \alpha_k p_k \tag{1.3}$$

where we have to choose opportunely, at every step, a direction p_k and a step length α_k.
Most algorithms choose as p_k a direction along which the objective function decreases, namely a *descent direction*:

Definition 1.2.1. *The vector p_k is a **descent direction** for f in x_k if exists $\bar{\alpha}_k > 0$ such that $f(x_k + \alpha_k p_k) < f(x_k)$ for all $\alpha_k \in (0, \bar{\alpha}_k]$.*

Taylor's theorem gives a condition for a vector p_k to be a descent direction:

Proposition 1.2.1. If results that

$$p_k^T \nabla f(x_k) < 0 \tag{1.4}$$

then p_k is a descent direction for f in x_k.

Proof. In fact, from the (1.1), we have that

$$f(x_k + \varepsilon p_k) = f(x_k) + \varepsilon p_k^T \nabla f_k + O(\varepsilon^2)$$

and it follows that $f(x_k + \varepsilon p_k) < f(x_k)$ for all positive but sufficiently small values of ε. □

Since (1.4) can be expressed as

$$p_k^T \nabla f(x_k) = \|p_k\| \, \|\nabla f(x_k)\| \cos\theta_k < 0$$

where θ_k is the angle between p_k and $\nabla f(x_k)$, in order to have a descent direction it is sufficient that the vector p_k forms an acute angle respect to the gradient $\nabla f(x_k)$, provided that the step length is sufficiently small.

Once chosen the direction, now we would like to choose the step length α_k without high costs. Since it is known that it is not sufficient just that $f(x_k + \alpha_k p_k) \leq f(x_k)$, we should make a step length choice such that we get a substantial reduction of f. The ideal choice would be the global minimizer of the univariate function defined by

$$\arg\min_{\alpha>0} \; \phi(\alpha) = \arg\min_{\alpha>0} \; f(x_k + \alpha p_k)$$

but, in general, it is too expensive to identify this value, because we should evaluate the objective function f and eventually the gradient ∇f a lot of times.

A common inexact line search condition that ensures a sufficient decrease of the objective function f is the *Armijo condition*:

$$f(x_k + \alpha_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k,$$

for some constant $c_1 \in (0, 1)$ and, in practice, usually $c_1 = 10^{-4}$. As we can see in Figure 1.1, this condition states that the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^T p_k$. But the sufficient decrease condition cannot be sufficient alone, since one could take too small values of α without making reasonable progress. So,

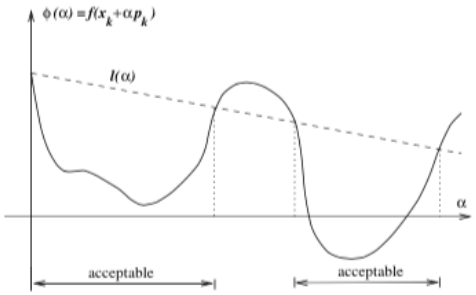


Figure 1.1: Armijo condition.

in order to reject too small values of α_k, we introduce a second requirement, called the *curvature condition*:

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k,$$

for some constant $c_2 \in (c_1, 1)$.

In the practice, instead of implementing both the sufficient decrease condition and the curvature condition (known collectively as *Wolfe conditions*), in order to avoid choosing too small values of α_k we use a so-called *backtracking* approach. In this procedure, the idea is to choose the biggest value of α_k that satisfies the Armijo condition. More precisely, the initial step length $\bar{\alpha}$ is chosen to be 1 and it is multiplied by a factor $\rho < 1$ (for example, $\rho = 1/2$) as long as that α value satisfies the Armijo condition. The backtracking approach algorithm can be written in pseudocode as in Algorithm 1.

Algorithm 1: backtracking algorithm

Choose $\bar{\alpha} > 0,\ \rho \in (0, 1)$. Set $\alpha \leftarrow \bar{\alpha}$;
while $f(x_k + \alpha_k) \geq f(x_k) + c_1 \alpha \nabla f_k^T p_k$ **do**
  $\alpha \leftarrow \rho\alpha$
end
$\alpha_k = \alpha$

1.2.2 Trust region algorithms

In the second algorithmic strategy, known as trust region, a model function m_k is constructed such that its behaviour near the current point x_k is similar to that of the objective function f. Since the model m_k may not be a good approximation of f when x is far from x_k, we restrict the search for a minimizer of m_k to some region around x_k, that is we find the candidate step p by approximately solving the subproblem:

$$\arg\min_p m_k(x_k + p), \quad \text{where } x_k + p \text{ lies in the trust region.} \tag{1.5}$$

If the candidate solution does not produce a sufficient decrease in f, then the trust region is too large and so we go on shrinking it and resolving the problem again. Usually, the trust region is a ball defined by $\|p\|_2 \leq \Delta$ (where the scalar $\Delta > 0$ is called the trust-region radius) but elliptical or box-shaped trust regions may also be used.

The model in (1.5) is usually defined to be a quadratic function of the form

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B_k p$$

where f_k and ∇f_k are respectively the function and the gradient values at the point x_k while the matrix B_k can be either the Hessian $\nabla^2 f_k$ or some approximation to it.

In conclusion, we note that in a sense the line search and trust-region approaches differ in the order in which they choose the direction and distance of moving to the next iterate. In fact, the line search strategy starts by fixing the direction p_k and then identifying an appropriate step length α_k. Instead, in trust region strategy we first choose a maximum distance, namely the trust-region radius Δ_k, and then find a direction and step that attain the

best improvement possible. If this step proves to be unsatisfactory, we reduce the distance measure Δ_k and try again.

1.2.3 Conjugate gradient method

We present in this section the conjugate gradient method, that is useful both in solving large linear systems of equations and solving nonlinear optimization problems. In fact, the conjugate gradient method can be formulated as an iterative method for solving a linear system of equations

$$Ax = b$$

where A is an $n \times n$ symmetric positive definite matrix or, equivalently, as the following minimization problem

$$\arg\min \phi(x) := \frac{1}{2}x^T Ax - b^T x. \tag{1.6}$$

namely a technique for minimizing convex quadratic functions. The two problems, for the Theorem 1.1.5, has the same solution $x^* = A^{-1}b$.

From (1.6) we have that the gradient of ϕ is equal to the residual of the linear system, namely

$$\nabla\phi(x) = Ax - b =: r(x)$$

and so, for $x = x_k$ we have

$$r_k = Ax_k - b$$

One of the remarkable properties of the conjugate gradient method is its

ability to generate in a very economically way a set of *conjugate* vectors:

Definition 1.2.2. *A set of nonzero vectors $p_0, p_1, \ldots, p_l$ is said to be conjugate with respect to the symmetric positive definite matrix A if*

$$p_i^T A p_j = 0, \quad \text{for all } i \neq j.$$

and it is easy to show that any set of vectors satisfying this property is also linearly independent.

Let us consider now a set of conjugate directions $\{p_0, p_1, \ldots, p_{n-1}\}$ and let us generate the sequence $\{x_k\}$ by setting

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \tag{1.7}$$

where α_k is the one-dimensional minimizer of the quadratic function ϕ along $x_k + \alpha_k$. Thanks to the next theorem, the importance of conjugacy lies in the fact that we can minimize (1.6) in n steps by minimizing it successively along the individual directions in a conjugate set:

Theorem 1.2.1. *For any $x_0 \in \mathbb{R}^n$ the sequence* (1.7) *generated by the conjugate direction algorithm converges to the solution $x^* = A^{-1}b$ in at most n steps. Moreover, the residuals are such that*

$$r_k^T p_i = 0, \ i = 0, 1, \ldots, k-1$$

and x_k is the minimizer of (1.6) *over the set*

$$\{x | x = x_0 + span\{p_0, p_1, \ldots, p_{k-1}\}\}.$$

A set of n $A-$ conjugate vectors always exists, and it is the set of the

eigenvectors of A, but it is not cheap to calculate them exactly. In the conjugate gradient method we chose the conjugate directions computing a new vector p_k by using only the previous vector p_{k-1}. It does not need to know all the previous elements $\{p_0, p_1, \ldots, p_{k-2}\}$, so the method requires small storage and computation, and p_k results automatically conjugate to these vectors. This statement is proved by the following theorem:

Theorem 1.2.2. *The iterative method given by*

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

where

$$\begin{aligned} p_k &= -\nabla f(x_k) + \beta_{k-1} p_{k-1}, \quad p_0 = -\nabla f(x_0) \\ \alpha_k &= -\frac{(\nabla f(x_k))^T p_k}{p_k^T A p_k} \\ \beta_{k-1} &= \frac{(\nabla f(x_k))^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \end{aligned}$$

are such that $\{p_0, p_1, \ldots, p_{n-1}\}$ are a set of descent and conjugate and directions, so the sequence $\{x_k\}$ converges to the solution $x^* = A^{-1}b$ in at most n steps. Moreover, the residuals are such that

$$r_k^T r_j = 0, \quad j = 0, 1, \ldots, k-1, \quad k = 1, 2, \ldots, n-1.$$

In the practice, we can write a computationally cheap different version of the conjugate gradient method. This version has a different expression for the coefficients α_k and β_k:

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}, \qquad \beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

but has the same proprieties of the previous one. Since it has several computation advantages, this is the version that we implemented in our tests. The algorithm of conjugate gradient method expressed in pseudocode is Algorithm 2.

Algorithm 2: CG algorithm

Given x_0;
Set $r_0 \leftarrow Ax_0 - b,\ p_0 \leftarrow -r_0,\ k \leftarrow 0$;
while $r_k \neq 0$ **do**
  $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$;
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
  $r_{k+1} \leftarrow r_k + \alpha_k A p_k$;
  $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;
  $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$;
  $k \leftarrow k + 1$;
end

1.2.4 CGLS method

The CGLS method solve a linear least-squares problem that takes the form

$$\arg\min \phi(x) := \frac{1}{2} = \|Qx - c\|^2 = \frac{1}{2} x^T Q^T Q x - x^T Q^T c.$$

that is a quadratic form problem with $Q \in \mathbb{R}^{m \times n},\ m \geq n$. This problem has a solution x if and only if

$$Q^T Q x = Q^T c \tag{1.8}$$

namely if x is a solution of the normal equation system. If the coefficient matrix $Q^T Q$ is positive definite we can use the conjugate gradient method in order to solve (1.8). Now we present in Algorithm 3 the pseudocode to solve this kind of problem using the CG method.

Algorithm 3: CGLS algorithm

Given x_0;
Set $r_0 \leftarrow Qx_0,\ z_0 \leftarrow Q^T r_0,\ p_0 = z_0,\ k \leftarrow 0$;
while $r_k \neq 0$ **do**
  $w_k \leftarrow Qp_k$;
  $\alpha_k \leftarrow \frac{z_k^T z_k}{w_k^T w_k}$;
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
  $r_{k+1} \leftarrow r_k - \alpha_k w_k$;
  $z_{k+1} \leftarrow Q^T r_{k+1}$;
  $\beta_{k+1} \leftarrow \frac{z_{k+1}^T z_{k+1}}{z_k^T z_k}$;
  $p_{k+1} \leftarrow z_{k+1} + \beta_{k+1} p_k$;
  $k \leftarrow k + 1$;
end

1.3 Algorithms for nonlinear least-squares problems

In this paragraph we are going to introduce the nonlinear least-squares problem and highlight the advantages we face if the optimization problem is formulated in this way. Then we will present some algorithms to solve this kind of problem that will be implemented for our experiments.

In a typical least-squares problem the objective function f is written as

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x) \tag{1.9}$$

where $r_j : \mathbb{R}^n \longrightarrow \mathbb{R}$ are differentiable functions named *residuals* and we always suppose $m \geq n$.

Least-squares problems arise in many applications (chemistry, physics, finance, economy...) when we have to solve a data fitting problem, namely

to select values for the parameters that best match the constructed model to the observed behaviour of the system we want to study. More precisely, in this kind of problems the function of the form (1.9) measures the discrepancy between the constructed model values $\phi(x; t_j)$ and the observed data y_j, so the residuals can be expressed as

$$r_j(x) = \phi(x; t_j) - y_j$$

and the (1.9) as

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} [\phi(x; t_j) - y_j]^2$$

In order to highlight why if the objective function f is expressed by (1.9) it makes least-squares problems easier to solve than general unconstrained minimization problems, we first assemble together the components $r_j : \mathbb{R}^n \longrightarrow \mathbb{R}$ in a single residual vector $r : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ as follows:

$$r(x) = (r_1(x), r_2(x), \ldots, r_m(x))^T .$$

With this notation, we can express the derivatives of $f(x)$ in terms of the $m \times n$ Jacobian matrix $J(x)$, composed by the first partial derivatives of the residuals, namely

$$J(x) = \left[\frac{\partial r_j}{\partial x_i}\right]_{\substack{j=1,2,\ldots,m \\ i=1,2,\ldots,n}} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}$$

where each $\nabla r_j(x)$ is the gradient of the residual $r_j(x)$, $j = 1, 2, \ldots, m$. In this way, the gradient and the Hessian of f can be expressed by:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^T r(x), \tag{1.10}$$

$$\begin{aligned} \nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) = \\ &= J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x). \end{aligned} \tag{1.11}$$

In many applications, the first partial derivatives of the residuals and hence the Jacobian matrix $J(x)$ are relatively easy or inexpensive to calculate, so we can obtain the gradient $\nabla f(x)$ using the (1.10). Moreover, using $J(x)$ we are also able to calculate the first term $J(x)^T J(x)$ of the Hessian $\nabla^2 f(x)$ in the (1.11) without calculating any second derivatives of the residual functions r_j, and this is a big advantage of this least-squares problem formulation.

Moreover, the term $J(x)^T J(x)$ is often more important than the second summation term in (1.11), because near the solution either the residuals r_j are close to affine (and then the $\nabla^2 r_j(x)$ are relatively small) or the residuals r_j are small. Most algorithms for nonlinear least-squares exploit these structural properties of the Hessian.

In many models $\phi(x; t)$, and then the residuals $r_j(x)$, are *linear* functions of x and the problem is called a *linear* least-squares problem. In this case, we can write the residual vector as $r(x) = Jx - y$ for some matrix J and vector y, both independent of x, so that the objective function becomes

$$f(x) = \frac{1}{2} \|Jx - y\|^2 \tag{1.12}$$

with $y = r(0)$. Since for hypothesis $\nabla^2 r_j = 0$ for all $j = 1, 2, \ldots, m$, for the (1.10) and (1.11) we have

$$\nabla f(x) = J(x)^T(Jx - y), \qquad \nabla^2 f(x) = J(x)^T J(x)$$

It can be shown that, only for a linear least-squares problem, $f(x)$ in (1.12) is a convex function. So, thanks to Theorem 1.1.5, every point x^* such that $\nabla f(x^*) = 0$ will be a global minimizer. Then our problem now is led back to solve a linear system of equations that is:

$$J^T J x^* = J^T y \tag{1.13}$$

known as the *normal equations system* for (1.12).

But, since in the future we will have to solve a *nonlinear* least-squares problem, now we are going to present some useful methods to solve this kind of problem. All the following methods can be considered as *line search modified Newton's method*, because they all follow from the definition of the definition of the *Newton direction*.

1.3.1 Newton direction and modified Newton's methods

One of the most important search direction is the Newton direction. This direction derives from the second-order Taylor series approximation to $f(x_k + p)$, that is

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p =: m_k(p)$$

Assuming that $\nabla^2 f_k$ is positive definite, we obtain the Newton direction by finding the vector p that minimizes $m_k(p)$. Using the Theorem 1.1.4, by setting the derivative of $m_k(p)$ to zero we obtain the following explicit formula for the Newton direction:

$$p_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k \tag{1.14}$$

So, finding the Newton direction means to find the solution p_k^N of the linear system

$$\nabla^2 f_k \; p_k^N = -\nabla f_k \tag{1.15}$$

The Newton direction is reliable when the difference between the true function $f(x_k+p)$ and its quadratic model $m_k(p)$ is not too large and can be used in a line search method only when $\nabla^2 f_k$ is positive definite. In this case, for the (1.15) we have

$$\nabla^2 f_k^T \; p_k^N = -{p_k^N}^T \nabla^2 f_k \; p_k^N \leq -\sigma_k \|p_k^N\|^2 < 0$$

for some $\sigma_k > 0$. Therefore, unless the gradient ∇f_k and then p_k^N is zero, we have that the Newton direction is a descent direction.

But, far from the solution, it can happens that $\nabla^2 f_k$ is not positive definite and then the Newton direction may not even be defined, since $\nabla^2 f_k$ may not exist. Even when it is defined, it may not satisfy the descent property $\nabla^2 f_k^T \; p_k^N < 0$ that makes it a good direction. In these case, we can make some modifications in the Newton method. If, for example, there is a point at which the Hessian is not positive defined, we could replace it by a positive defined approximation of the Hessian just adding, for example, a diagonal matrix or a full matrix. All this methods are called *Newton's methods with*

Hessian modification or *modified Newton's methods.*

1.3.2 Gauss-Newton method

The Gauss–Newton method can be viewed as a modified Newton's method. In fact, instead of solving the system (1.15), keeping in mind all the observations we did about the advantages of the least-squares formulation and, in particular, about equation (1.11), the Gauss–Newton method is obtained making the approximation

$$\nabla^2 f_k \approx J_k^T J_k \tag{1.16}$$

and then, in order to find the direction p_k^{GN}, we have to solve the new system

$$J_k^T J_k \; p_k^{GN} = -\nabla f_k \tag{1.17}$$

As we said, this approximation has several advantages. Firstly, we save some computational time because we do not have to compute the individual residual Hessians $\nabla^2 r_j, \;\; j = 1, 2, \ldots, m$ that are needed in the second term of (1.11) and nowhere else. Secondly, in many situation the first term $J_k^T J_k$ dominates the second term (at least close to the solution x^*), so it makes a good approximation of the Newton's method.

A new third advantage is that whenever J_k has full rank and the gradient ∇f_k is nonzero, the direction p_k^{GN} is a descent direction for f, and then a good direction for a line search. In fact from (1.10) and (1.16) we have

$$(p_k^{GN})^T \nabla f_k = (p_k^{GN})^T J_k^T r_k = -(p_k^{GN})^T J_k^T J_k p_k^{GN} = -\|J_k p_k^{GN}\|^2 \leq 0$$

The final inequality is strict, unless $J_k p_k^{GN} = 0$. In this case, it follows from (1.16) and the full rank of J_k that $J_k^T r_k = \nabla f_k = 0$, namely x_k is a stationary

point.

Finally, the fourth advantage of Gauss–Newton method comes from the similarity between the equations (1.16) and the normal equations (1.13) for the linear least-squares problem. In fact, p_k^{GN} can be seen as the solution of the linear least-squares problem

$$\arg\min_p \frac{1}{2}\|J_k p + r_k\|^2 \tag{1.18}$$

and then it can be solved by applying linear least-squares algorithms to the subproblem (1.18).

1.3.3 Levenberg-Marquardt method

The Levenberg–Marquardt method can be obtained by using the same Hessian approximation (1.16) of the Gauss-Newton method, but replacing the line search with a trust-region strategy. Actually, the Levenberg–Marquardt method is sometimes considered to be the progenitor of the trust-region approach for general unconstrained optimization problems. The use of this type of strategy avoids one of the weaknesses of Gauss–Newton method, that is its behaviour when the Jacobian $J(x)$ is rank-deficient, or nearly so. Then, since the same Hessian approximation (1.16) are used in both cases, the local convergence properties of the two methods are similar.

In the Levenberg-Marquardt method we consider a spherical trust region and we have to solve at each iteration the subproblem

$$\arg\min_p \frac{1}{2}\|J_k p + r_k\|^2, \quad \text{with } \|p\| \leq \Delta_k \tag{1.19}$$

where Δ_k is the trust-region radius. In this way, we choose the trust regiorn model function (1.5) as

$$m_k(p) = \frac{1}{2}\|r_k\|^2 + p^T J_k^T r_k + \frac{1}{2} p^T J_k^T J_k p$$

This methods can be expressed by this idea: if the solution p_k^{GN} of the Gauss–Newton equations (1.17) lies strictly inside the trust region, namely $\|p_k^{GN}\| < \Delta_k$, then this step p_k^{GN} also solves the subproblem (1.19). If not, then there is a $\mu_k > 0$ such that the solution $p_k = p_k^{LM}$ of (1.19) satisfies $\|p_k\| = \Delta_k$ and

$$(J_k^T J_k + \mu_k I)p_k = -J_k^T r_k. \tag{1.20}$$

This claim is stated by the following lemma:

Lemma 1.3.1. *The vector p_k^{LM} is a solution of the trust-region subproblem* (1.19) *if and only if p_k^{LM} is feasible and there is a scalar $\mu_k \geq 0$ such that*

$$(J_k^T J_k + \mu_k I)p_k^{LM} = -J_k^T r_k \tag{1.21}$$

$$\mu_k(\Delta_k - \|p_k^{LM}\|) = 0 \tag{1.22}$$

The Levenberg-Marquardt method can be interpreted as solving the normal equations used in the Gauss-Newton method, but "shifted" by a scaled identity matrix. In this way, we convert the problem from having an ill-conditioned (or positive semidefinite) matrix $J_k^T J_k$ into a positive definite one, and then that the Levenberg-Marquardt method is well defined.

The size of the parameter μ_k usually influences both the direction and the length step, and we come back to the Gauss-Newton method for $\mu = 0$. The several versions of the Levenberg-Marquardt method differ from the way μ_k is chosen at every iteration. In our experiments, as we will see in the next

chapters, we have chosen some different criteria of updating of μ_k values. In general, the most efficient methods try to assign iteratively to μ_k a value such that (1.21) is satisfied and they are based on some particular factorizations of the matrix $J_k^T J_k + \mu_k I$.

1.3.4 Two L-BFGS methods

A subclass of the modified Newton's methods is that of the so-called *quasi-Newton methods*. If the standard Newton method needs to calculate the direction p_k as in (1.29), in a quasi-Newton method we try to calculate p_k by approximating or the inverse of the Hessian matrix $H_k := \nabla^2 f_k$ with a positive definite matrix

$$G_k \approx H_k^{-1}$$

such that G_{k+1} can be obtained from G_k in $O(n^2)$ steps, or by approximating directly the Hessian matrix H_k with a matrix

$$B_k \approx H_k$$

with the same properties and $G_k = H_k^{-1}$. In this way, the idea is that of obtaining the direction p_k as a result of the application of a certain linear operator to the gradient vector calculated in the current point:

$$p_k = -G_k \nabla f_k$$
$$p_k = -B_k^{-1} \nabla f_k.$$

The most popular quasi-Newton algorithm is the BFGS method and takes its name from its discoverers Broyden, Fletcher, Goldfarb and Shanno. One of the several versions of the BFGS methods consists on the approximation

of H_k with the update formula given by

$$B_{k+1} = B_k + \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{1.23}$$

where

$$s_k = x_{k+1} - x_k = \alpha_k p_k$$
$$y_k = \nabla f_{k+1} - \nabla f_k$$

and they are such that satisfy the *curvature condition*

$$s_k^T y_k > 0. \tag{1.24}$$

Since in many applications we have to solve such a large problem that we should need a lot of storage and time in order to store fully dense $n \times n$ approximations of Hessian matrices, various limited-memory methods have been proposed. One of these is the class of Limited-memory BFGS methods (or L-BFGS), which involve the BFGS method but need to store only a certain number (say, m) of the latest vector pairs (s_i, y_i), discarding the oldest ones.

In order to reach a practical algorithm belonging to L-BFGS methods, firstly we observe that the direct BFGS formula (1.23) can be expressed as

$$B_{k+1} = B_k - a_k a_k^T + b_k b_k^T \tag{1.25}$$

where the vectors a_k and b_k are defined by

$$a_k = \frac{B_k s_k}{\sqrt{s_k^T B_k s_k}}$$
$$b_k = \frac{y_k}{\sqrt{y_k^T s_k}}.$$

If we want to use a limited-memory approach, from (1.25) we could define an initial matrix B_k^0 and than at each iteration update the matrix according to the formula

$$B_k = B_k^0 + \sum_{i=k-m}^{k-1} \left[b_i b_i^T - a_i a_i^T \right]$$

where, for example, B_k^0 can be taken as $\gamma_k^{-1} I$ with I the identity matrix and $\gamma_k > 0$ constant. Often, in practice, γ_k is chosen as

$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}. \tag{1.26}$$

In this way, the vector pairs $(a_i, b_i),\ i = k-m, k-m+1, \ldots, k-1$ can be recovered from the known stored vector pairs $(s_i, y_i),\ i = k-m, k-m+1, \ldots, k-1$ by the unrolling procedure written in Algorithm 4.

In our future experiments we would like to use the L-BFGS formula in

Algorithm 4: Unrolling the BFGS formula

for $i = k-m, k-m+1, \ldots, k-1$ **do**
- $b_i \leftarrow \frac{y_i}{\sqrt{y_i^T s_i}}$;
- $a_i \leftarrow B_k^0 s_i + \sum_{j=k-m}^{i-1} \left[(b_j^T s_i) b_j - (a_j^T s_i) a_j \right]$;
- $a_i \leftarrow \frac{a_i}{\sqrt{s_i^T a_i}}$;

end

order to solve a regularized problem that takes the form of the Levenberg-Marquardt method, namely expressed as

$$(H_k + \mu_k I)p_k = -\nabla f_k. \tag{1.27}$$

Taking this in mind, we could prove that, once known $(a_i, b_i),\ i = k-m, k-m+1, \ldots, k-1$, Algorithm 5, by using a recursion formula, is able to find the inverse of $B+\sigma I$ or, more precisely, solves systems of the form $(B+\sigma I)x = z$ for any z (for the proof, see [2]).

So, putting all together, now we are able to write the L-BFGS algorithm

Algorithm 5: Recursion formula to compute $r = (B + \sigma I)^{-1}z$

$r \leftarrow \frac{z}{\gamma_{k+1}^{-1}+\sigma}$;
for $j = 0, \ldots, k$ **do**
  if j *even* **then**
    $c \leftarrow a_{\frac{j}{2}}$;
  else
    $c \leftarrow b_{\frac{j-1}{2}}$;
  end
  $p_j \leftarrow \frac{c}{\gamma_{k+1}^{-1}+\sigma}$;
  for $i = 0, \ldots, j-1$ **do**
    $p_j \leftarrow p_j + (-1)^i v_i (p_i^T c) p_i$;
  end
  $v_j \leftarrow (1 + (-1)^j p_j^T c)^{-1}$;
  $r \leftarrow r + (-1)^j v_j (p_j^T z) p_j$;
end

to solve the problem (1.27). The procedure is expressed in Algorithm 6.

Another strategy can be that of constructing from f a new function $f^{(\mu)}$ defined as

$$f^{(\mu)}(x) := f(x) + \frac{\mu}{2}\|x\|^2.$$

Algorithm 6: L-BFGS method

Given starting point x_0, m, convergence tolerance $tol > 0$, μ_k for all k ;
$k \leftarrow 0$;
while $\|\nabla f_k \neq 0\|$ **do**
  Check the curvature condition (1.24) ;
  Compute γ_k as (1.26) ;
  Compute (a, b) using Algorithm 4 ;
  Compute search direction p_k using Algorithm 5 with
  $B = H_k,\ \sigma = \mu_k,\ z = -\nabla f_k$;
  Check p_k is a descent direction ;
  Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from the backtracking procedure given by Algorithm 1 ;
  Define $s_k = x_{k+1} - x_k,\ y_k = \nabla f_{k+1} - \nabla f_k$ and store only the m last (s_i, y_i) pairs ;
  $k \leftarrow k + 1$;
end

Algorithm 7: Two-loop recursion to compute $r = B_k^{-1} z$

$q \leftarrow z$;
for $i = k - 1, \ldots, 0$ **do**
  $\alpha_i \leftarrow \frac{s_i^T q}{y_i^T s_i}$;
  $q \leftarrow q - \alpha_i y_i$;
end
$r \leftarrow (B_k^0)^{-1} q$;
for $i = 0, \ldots, k - 1$ **do**
  $\beta \leftarrow \frac{y_i^T r}{y_i^T s_i}$;
  $r \leftarrow r + (\alpha_i - \beta) s_i$;
end

Algorithm 8: L-BFGS method with $f^{(\mu)}$

Given starting point x_0, m, convergence tolerance $tol > 0$, μ_k for all k ;
$k \leftarrow 0$;
while $\|\nabla f_k \neq 0\|$ **do**
  Check the curvature condition (1.24) ;
  Compute γ_k as (1.26) ;
  Compute search direction p_k using Algorithm 7 with
  $B_k = H_k^{(\mu)},\ z = -\nabla f_k$;
  Check p_k is a descent direction ;
  Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from the backtracking procedure given by Algorithm 1 ;
  Define $s_k = x_{k+1} - x_k,\ y_k^{(\mu)} = \nabla f_{k+1}^{(\mu)} - \nabla f_k^{(\mu)}$ and store only the m last $(s_i, y_i^{(\mu)})$ pairs ;
  $k \leftarrow k + 1$;
end

Thanks to $f^{(\mu)}$, it is easy to prove that, naming $H_k^{(\mu)}(x) := \nabla^2 f_k^{(\mu)}(x)$, equation (1.27) can be rewritten as

$$H_k^{(\mu)} p_k = -\nabla f_k.$$

For this goal Algorithm 7 turns out to be very useful because, by using a two-loop recursion formula, it is able to find an approximation of $\left(H_k^{(\mu)}\right)^{-1}$ or, more precisely, solves systems of the form $B_k r = z$ for any z.
Then, via Algorithm 7, Algorithm 8 now offers another strategy in order to solve the problem (1.27). In addition to what we have already said, we note that this algorithm differs to the previous one because, instead of saving (s_i, y_i), $(s_i, y_i^{(\mu)})$ pairs are saved, where $y_k^{(\mu)} = \nabla f_{k+1}^{(\mu)} - \nabla f_k^{(\mu)}$.

1.4 Central-difference formula

In the final section of this chapter we introduce the way we will approximate the gradient in our experiments. An approximation to the gradient vector $\nabla f(x)$ can be obtained by evaluating the function f at $(n+1)$ points and performing some elementary arithmetic. An accurate approximation to the derivative can be obtained by using the *central-difference* formula, defined as

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f - (x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon}$$

and it requires to evaluate f at the points x and $x \pm \varepsilon e_i,\ i = 1, 2, \ldots, n$ for a total of $2n + 1$ points. The central difference approximation comes directly from Taylor's theorem 1.1.1. If the second derivatives of f exist and are Lipschitz continuous, we have from (1.1) that for some $t \in (0, 1)$

$$\begin{aligned} f(x+p) &= f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp) p \\ &= f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p + O(\|p\|^3). \end{aligned}$$

By setting respectively $p = \varepsilon e_i$ and $p = -\varepsilon e_i$ we get

$$f(x + \varepsilon e_i) = f(x) + \varepsilon \frac{\partial f}{\partial x_i} + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2} + O(\varepsilon^3) \tag{1.28}$$

$$f(x - \varepsilon e_i) = f(x) - \varepsilon \frac{\partial f}{\partial x_i} + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2} + O(\varepsilon^3) \tag{1.29}$$

By subtracting (1.29) from (1.28) and dividing by 2ε we obtain

$$\frac{\partial f}{\partial x_i}(x) = \frac{f(x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon} + O(\varepsilon^2) \tag{1.30}$$

that is the approximation of the $i-$th component of the gradient.

Chapter 2

Tomography and mathematical model

In this chapter our goal is to introduce some data acquisition techniques in the field of *Medical imaging*, which will be the topic of our work. After a short introduction to the Computerized Tomography (CT), we will focus on Digital Tomosynthesis technique. Since we will have to solve a high-dimensional ill-posed problem, we will need to present a physical model for the formation of the image, that will be multimaterial and polyenergetic. When the problem will be formulated in terms of non linear least squares minimization, its ill-position will lead us to the need to find a solution iteratively, after that some type of regularization is applied. Finally, we will show an achievable gradient vector approximation that appears in the numerical methods introduced in Chapter 1.

2.1 Computerized Tomography

Technological improvement that is passing through modern society revolutionized medical world too: the presence of more and more sophisticated devices together with radioactivity knowledge has gradually put in the foreground the field of the *medical radiology*, that is the medicine branch which studies images production and visualization for a diagnostic and therapeutic aim.

The main radiography idea consists in going back to the features of the organ we want to examine (unknown three-dimensional object) by watching a plate (two-dimensional image). This is possible thanks to the employment of X-rays that, generated from a radiogenic tube, pass through the patient; X-rays, according to the different materials' consistencies they hit, will meet more or less resistance and are in part attenuated and in part absorbed. In the other part of the tube some detectors absorb the passed through X-rays, and data, elaborated by a computer, permit the image formation. In this way, for example, crashing with a dense tissue like the bone tissue, the ray will be absorbed and the radiography will appear white; on the contrary, for less dense tissues or for the air the radiography will be dark-coloured.

Formulated in this way, the technique presents two limits: the first one is that two similar-consistence tissues (typically the softest ones) are bad observable; the second one is about the loss of information that we get by describing a three-dimensional object with a two-dimensional object. These two problems have been solved by the introduction of the Computerized Tomography (CT), known also like X-ray Computed Tomography (X-ray CT), since it permitted to distinguish similar absorption-capability structures from each other and to separate overlapping structures.

This diagnostic imaging methodology used for several decades consists in

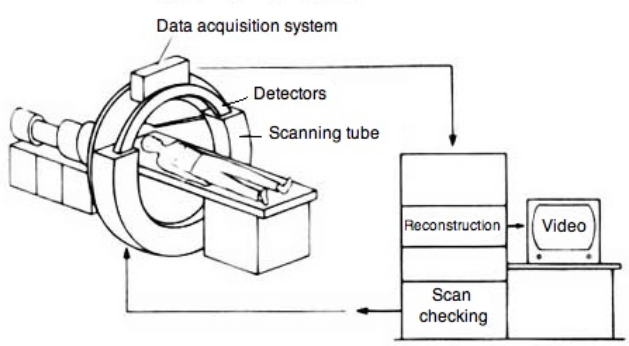
the representation of a three-dimensional body through a series of sections (*tomography*) along transverse planes, where the real acquired-data elaboration is made happen with the help of a computer (*computerized*). After the discovery of the X-rays, the advent of the Computerized Tomography can be considered one of the most important inventions in the medical radiology field, as its employment allowed to identify lesions previously difficult to see. Since via dedicated software CT permits also to realize a three-dimensional elaboration of acquired images, it runs for an optimal technique for the study of complex anatomical structures too and it is nowadays a very used exam, especially in presurgical diagnosis.

Compared to conventional radiography, Computerized Tomography presents considerable advantages, first of all it allows the localization of deep structures. Moreover, CT permits a better details view, since it is able to avoid distortions due to different planes overlapping, which are responsible of the *false positive* or *false negative* appearance. Finally, CT has the advantage to increase the contrast between similar local structures.

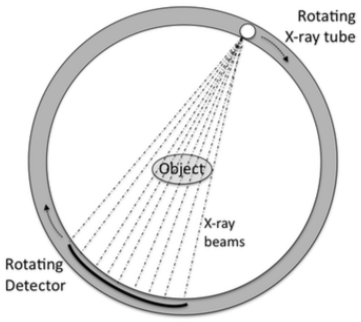
In this work we do not intend to show in detail the precise CT working process or the several CT machines evolutions but we restrict to describe, with the help of Figure 2.1, a typical CT machine functioning, without going into detail.

Patient table slides into the scan tube where a X-rays emitter rotates all around him, releasing radiations at determined angles; on the opposite side some detectors collect the rays attenuated by the patient's body. The table is allowed to slide very precisely, so that for each detector's position and angle it is possible to record the different section of the object we want to analyse. The data set recorded by detectors together with the table positions and angles values are collected and transmitted to a computer, where are

elaborated and transformed to images.



(a) Patient enters the scan tunnel where detectors acquire data; a computer elaborates them and creates the image.



(b) X-rays emitter rotates forming an angle of 360°.

Figure 2.1: Geometry of a typical CT machine.

2.2 Digital Tomosynthesis

If, as we said, Computerized Tomography has been a great step forward in the medical imaging field, instead we have to underline that it has some limits too. The biggest one is that this technique permits to acquire only one section at a time, so that this process has to be repeated several times, increasing in this way the data elaboration time and the exposition of the patient to radiations. All this makes this technique not always efficient, especially when patients are particularly sensible subjects like babies or if it has to be applied to particularly delicate organs, such as the breast or the chest.

In order to avoid this kind of problems, recently a new medical research technique has been introduced, the **Digital Tomosyntesis**, which undertakes to reconstruct (*synthesis*) a three-dimensional object from a limited number of sections (*tomograms*) with the support of a computer (*digital*).

This technique consists in providing some multiple projections of the object from a limited number of angles chosen from a fixed angular diameter in order to obtain a three-dimensional pseudo-representation of the object. In the case of the chest (Figure 2.2a), this is possible thanks to the rotation of the X-rays emitter along an arc in front of the patient; instead, in Figure 2.2b the emitter rotates over the compressed breast. Since the emitter covers a limited angle range (typically less than 40 degrees), Digital Tomosyntesis reduces the exposition of the patient to the radiations considerably. Moreover, since this is a volumetric reconstruction technique, the false positives and negatives emergence is limited.

As volumetric 3D object reconstruction has to manage millions of pixels resulting from the projections of billions of voxels which compose the object, this reconstruction technique is a large-scale problem that involves high computational and storage costs. Then, in order to dealing with such a inverse large-scale problem, usually a simplification of the physical model of acquiring data is needed. A first kind of simplification can be considering a monoenergetic X-ray beam, in order to reduce the problem's degrees of freedom and the computational complexity in the calculation of the solution. But these hypothesis have led to the appearance of artifacts in the solution, such as the *beam hardening* phenomenon. This phenomenon is due to the fact that when a radiation passes through a tissue, the beam's mean energy increases, since low-energy photons are in general more absorbed compared with high-energy ones. So the detector registers an exit radiation that has a higher mean energy than entrance one, and this causes a darkening of the image, especially in the central part (*cup artifact*). The presence of that artifacts, which descend from the hypothesis on the monoenergetic X-ray beam nature, can be corrected with post-processing operations.

What we undertake to do in this work is to solve a problem which comes from a Digital Tomosynthesis technique employment applied to a breast exposed to a polyenergetic X-ray beam, in order to avoid some artifacts like the beam hardening and to get quite good results without necessarily making any post-processing to the images.

(a) Chest Tomosynthesis. (b) Brest Tomosynthesis.

Figure 2.2: Device outline for the image acquisition for the Digital Tomosynthesis technique.

2.3 Inverse problem

In the imaging field, where the goal is to reconstruct an image from certain indirect measurements and corrupted by noise, algorithms try to solve an inverse large-scale problem that takes the form

$$b = K(X)s + \eta \tag{2.1}$$

where the vector b represents data measurements, the matrix K depends on unknown vector X while the vector η represents the noise component in data measurements.

The choice of the model used to describe the image formation process

defines how the matrix $K(X)$ is constructed and the nature of vector η, that can represent an error due to an electronic or dispersive effect noise. The chosen model will be more o less complex depending on the accuracy level we want to describe the process concerned with, and that choice will influence the implementation of the algorithm intended to solve the problem.

Once that b, s, η and how matrix $K(X)$ is constructed from X are known, then the goal will be to trace back to the unknown vector X, or at least to an approximation of it. We note that X does not necessarily represent the object itself, but it can represent (and that will be our case) a certain kind of information that characterizes the object, such as a density or another physical quantity in general.

What we want to do in this work is not to present and study the main models that can be found in literature, but instead to analyse particularly the **polyenergetic and multimaterial model for the breast image in Digital Tomosynthesis** presented in [3], where the model in question takes into account of the polyenergetic X-ray nature and the multimaterial composition of the object.

2.4 A first mathematical model

As we announced in the previous paragraph, we are going to show the image reconstruction model proposed in [3] and that will be discussed in this work. In order to do so, we start to consider a simpler model presented in [4] and than we will move from it to reach the real model that is the object of our experiments.

The starting point is the Beer's law, an expression which relates measured data, namely $b_i^{(\theta)}$, with the ability that the object has to absorbe or disperse

the quantity of the radiations that pass through it. More precisely, Beer's law states that

$$b_i^{(\theta)} = \int_{\varepsilon} s(\varepsilon) e^{-\int_{L_\theta} \mu(\vec{x},\varepsilon) dl} d\varepsilon + \eta_i^{(\theta)}, \qquad \begin{matrix} i = 1, 2, \ldots, N_p, \\ \theta = 1, 2, \ldots, N_\theta \end{matrix} \tag{2.2}$$

where:

- ▷ $b_i^{(\theta)}$ represents the quantity measured by the i-th pixel of a digital detector hit by a X-rays beam with entrance angle θ;
- ▷ N_p is the number of pixels of the digital rilevator hit by X-rays (it usually runs to about one million);
- ▷ N_θ is the number of angles covered by the radiations emitter (generally in a Tomosynthesis system we have $15 \leq N_\theta \leq 30$);
- ▷ ε represents the energy spectrum of X-rays source. A precise estimate of the X-rays distributions can be obtained by using known spectral models while calibration measurements can be obtained by measuring the X-rays transmission through objects which have well known dimensions, densities and materials.
- ▷ $s(\varepsilon)$ is the energy fluency, namely the product of the X-ray energy with the number or incident photons at that energy level;
- ▷ L_θ is the line through which X-rays beam with an inclination θ passes through the object;
- ▷ $\mu(\vec{x}, \varepsilon)$ is the linear attenuation coefficient, that depends on the X-rays beam energy and the object's material at the position $\vec{x}$. We highlight that in general lower-level energies are attenuated mainly if referred

to higher-level ones and that denser materials (such as bone tissue) attenuate mainly respect to less dense ones.

- $\eta_i^{(\theta)}$ represents the noise component observed by the i-th pixel respect to the angle θ; this contribution can include X-rays dispersion and electronic noise. The information about additional noise can be estimated with pre-processing or calibration operations.

From the discretization of (2.2) we obtain the discrete model of the formation of the image

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(-\sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \mu_{l,\varepsilon}\right) + \eta_i^{(\theta)}, \qquad \begin{matrix} i = 1, 2, \ldots, N_p, \\ \theta = 1, 2, \ldots, N_\theta \end{matrix} \tag{2.3}$$

where:

- N_v is the number of voxels after the object discretization (it usually runs to about one billion);
- N_ε is the number of discrete energy levels (since N_v is very high, it result $N_\varepsilon \ll N_v$);
- $a_{i,l}^{(\theta)}$ is the length of the ray that passes through the voxel l with a incidence angle θ, contributing to the i-th pixel.

Naming $A^{(\theta)} = [a_{i,l}^{(\theta)}]_{i,l}$ and $M = [\mu_{l,\varepsilon}]_{l,\varepsilon}$, the (2.3) can be expressed in a matrix format as

$$b^{(\theta)} = \exp(-A^{(\theta)} M)s + \eta^{(\theta)}, \quad \theta = 1, 2, \ldots, N_\theta$$

where the exponential function has to be interpreted applied to the matrix $-A^{(\theta)}M$ element by element. Then, considering all the possible values of the

incidence angle θ, the discrete model can be formulated as

$$b = \exp(-AM)s + \eta \tag{2.4}$$

with

$$b = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(N_\theta)} \end{bmatrix}, \quad A = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(N_\theta)} \end{bmatrix} \text{ and } \eta = \begin{bmatrix} \eta^{(1)} \\ \eta^{(2)} \\ \vdots \\ \eta^{(N_\theta)} \end{bmatrix}.$$

where the quantities which appear in (2.4) are such that:

- ▷ b is a vector of dimensions $(N_p \cdot N_\theta) \times 1$, since it is made up of N_θ blocks of dimensions $N_p \times 1$;
- ▷ A is a matrix of dimensions $(N_p \cdot N_\theta) \times N_v$, made up of N_θ blocks of dimensions $N_p \times N_v$;
- ▷ M is a matrix of dimensions $N_v \times N_\varepsilon$;
- ▷ s is a vector of dimensions $N_\varepsilon \times 1$;
- ▷ η is a vector of dimensions $(N_p \cdot N_\theta) \times 1$, made up of N_θ blocks of dimensions $N_p \times 1$.

Then we highlight that our inverse problem as in (2.1)

$$b = K(X)s + \eta$$

turns now into the form as in (2.4)

$$b = \exp(-AM)s + \eta$$

where

$$K(X) = \exp(-AM), \quad X = M.$$

The known data of the problem are the data vector b, the raytracing matrix A and fluency vector s (as we will see later, we will consider the noise component η absorbed into the data vector b). Our goal will be to find M, that is the matrix that represents a certain physical quantity that will allow us to reconstruct the object. In our case the physical quantity in question represents the linear attenuation coefficients, or rather the mass attenuation coefficients, as we will have an opportunity to explain better in the next paragraph.

Before passing to the real description of the model studied in this work, we highlight that the presence of the exponential function in equation (2.4) makes this inverse problem non linear, and this takes shape in the need of solve a computationally harder problem. In literature there are several methods, which we do not deal with in this work, that, making some simplifications in the physical model (like, for example, considering just monoenergetic X-rays) permit us to lead back to the solution of a linear inverse problem, that is more easily resolvable. But these simplifications have negative consequences, since they usually lead to the appearance of artifacts in the solution.

2.5 A polyenergetic multimaterial model

Now we introduce the description of the polyenergetic multimaterial model for the breast reconstructed image in Digital Tomosynthesis presented in [3]. This model is characterized by two main aspects, namely it takes account of the radiation polyenergetic nature and the variety of materials making up the object to reconstruct. Traditionally, in fact, it has been assumed for

simplicity that X-rays were emitted all at the same energy level and this simplification led to the appearance of *beam hardening* type artifacts in the reconstructed image. The choice to consider the object composed of more than one material (like, for example, glandular tissue, bone tissue and microcalcifications) stems from the fact of creating a more flexible model, that is able to extend this image reconstruction technique to other body components too.

The model description that we want to illustrate in this thesis is linked to a previous work [4]. The starting point of the construction of that model is the discretization of the Beer's law that we introduced in the previous paragraph (equation (2.3))

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(-\sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \mu_{l,\varepsilon}\right) + \eta_i^{(\theta)}, \qquad \begin{array}{l} i = 1, 2, \ldots, N_p, \\ \theta = 1, 2, \ldots, N_\theta. \end{array}$$

Taking into account the multimaterial nature of the object, each linear attenuation coefficient can be expressed as a weighted average of the linear attenuation coefficients of the single materials, namely:

$$\mu_{l,\varepsilon} = \sum_{m=1}^{N_m} w_{l,m} c_{m,\varepsilon} \tag{2.5}$$

where:

- ▷ N_m is the number of material making up the object;
- ▷ $c_{m,\varepsilon}$ is the linear attenuation coefficient of the m-th material at energy level ε;
- ▷ $w_{l,m}$ is the unknown weight of the m-th material in the l-th voxel of the object.

At this point the unknowns are the weights $w_{l,m}$, that have to take account of constraint of adding together to 1 in each voxel as materials change:

$$\sum_{m=1}^{N_m} w_{l,m} = 1, \quad l = 1, 2, \ldots, N_v. \tag{2.6}$$

Using (2.5), (2.3) becomes

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(- \sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \sum_{m=1}^{N_m} w_{l,m} c_{m,\varepsilon} \right) + \eta_i^{(\theta)}, \qquad \begin{matrix} i = 1, 2, \ldots, N_p, \\ \theta = 1, 2, \ldots, N_\theta \end{matrix} \tag{2.7}$$

Now, starting from these considerations, we move to the real model we undertake to study. At this point let us consider a new physical quantity, that is the mass attenuation coefficients, which are linked to the linear attenuation coefficients by this expression:

$$\mu_{l,\varepsilon} = \rho_l \delta_{l,\varepsilon} \tag{2.8}$$

where:

- ▷ $\delta_{l,\varepsilon}$ is the mass attenuation coefficient of the l-th voxel at energy level ε;
- ▷ ρ_l represents the density of composition of the materials in the l-th voxel.

In much the same way as we did in (2.5), we express each mass attenuation coefficient as a weighted average of the mass attenuation coefficients of the single materials, namely:

$$\delta_{l,\varepsilon} = \sum_{m=1}^{N_m} \widetilde{w}_{l,m} d_{m,\varepsilon} \tag{2.9}$$

where:

- ▷ $d_{m,\varepsilon}$ is the mass attenuation coefficient of the m-th material at energy level ε;
- ▷ $\widetilde{w}_{l,m}$ is the unknown weight of the m-th material in the l-th voxel of the object.

Now the unknowns are the weights $\widetilde{w}_{l,m}$, which have to be such that

$$\sum_{m=1}^{N_m} \widetilde{w}_{l,m} = 1, \quad l = 1, 2, \ldots, N_v. \tag{2.10}$$

Replacing (2.8) and (2.9) in (2.3), we get:

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(- \sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \rho_l \delta_{l,\varepsilon} \right) + \eta_i^{(\theta)}, \tag{2.11}$$

$$= \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(- \sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \rho_l \sum_{m=1}^{N_m} \widetilde{w}_{l,m} d_{m,\varepsilon} \right) + \eta_i^{(\theta)}, \tag{2.12}$$

$$i = 1, 2, \ldots, N_p, \quad \theta = 1, 2, \ldots, N_\theta.$$

Now we prove now an expression for the densities ρ_l:

Proposition 2.5.1. We have

$$\rho_l = \frac{1}{\displaystyle\sum_{m=1}^{N_m} \frac{\widetilde{w}_{l,m}}{r_m}}, \quad l = 1, 2, \ldots, N_v \tag{2.13}$$

where r_m is the density of the m-th material.

Proof. Since the mass attenuation coefficient of the m-th material $d_{m,\varepsilon}$ verifies the relation

$$c_{m,\varepsilon} = r_m d_{m,\varepsilon}, \tag{2.14}$$

obtaining $d_{m,\varepsilon}$ from (2.14) and substituting it in (2.12) we get

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(-\sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \rho_l \sum_{m=1}^{N_m} \widetilde{w}_{l,m} \frac{c_{m,\varepsilon}}{r_m} \right) + \eta_i^{(\theta)}, \qquad \begin{array}{l} i = 1, 2, \ldots, N_p, \\ \theta = 1, 2, \ldots, N_\theta. \end{array} \tag{2.15}$$

By comparing (2.15) with (2.7) we have

$$\rho_l \widetilde{w}_{l,m} \frac{1}{r_m} = w_{l,m}.$$

Hence, adding on m and using (2.6), we obtain

$$\rho_l \sum_{m=1}^{N_m} \frac{\widetilde{w}_{l,m}}{r_m} = \sum_{m=1}^{N_m} w_{l,m} = 1$$

and then the thesis. □

So, substituting (2.13) in (2.12) we get:

$$b_i^{(\theta)} = \sum_{\varepsilon=1}^{N_\varepsilon} s_\varepsilon \exp\left(-\sum_{l=1}^{N_v} a_{i,l}^{(\theta)} \frac{\displaystyle\sum_{m=1}^{N_m} \widetilde{w}_{l,m} d_{m,\varepsilon}}{\displaystyle\sum_{m=1}^{N_m} \frac{\widetilde{w}_{l,m}}{r_m}} \right) + \eta_i^{(\theta)}, \tag{2.16}$$

$$i = 1, 2, \ldots, N_p, \quad \theta = 1, 2, \ldots, N_\theta.$$

The equation (2.16) can be expressed in matrix format as

$$b = \exp\left(-A \left(\frac{\widetilde{W} D}{\widetilde{W} R} \right) \right) s + \eta \tag{2.17}$$

where

- ▷ $\widetilde{W}$ is the matrix of unknown weights $\widetilde{w}_{l,m}$ and dimensions $N_v \times N_m$;

▷ D is the matrix of dimensions $N_m \times N_\varepsilon$ and elements $[D]_{m,\varepsilon} = d_{m,\varepsilon}$;

▷ R is the matrix of dimensions $N_m \times N_\varepsilon$ and elements $[R]_{m,\varepsilon} = \dfrac{1}{r_m}$

and the exponential and division operations are applied to matrices element by element. Substituting to b the term $\tilde{b} = b - \eta$, where the noise term is absorbed, (2.17)

$$\tilde{b} = \exp\left(-A\left(\frac{\widetilde{W}D}{\widetilde{W}R}\right)\right)s.$$

What we expect to find is a weight combination $\widetilde{W}$ that, from all the available ones, is such that the distance (measured in some kind of norm) between observed data and data predicted by constructed model is small. More formally: if $\tilde{r} : \mathbb{R}^{N_v \cdot N_m} \longrightarrow \mathbb{R}^{N_p \cdot N_\theta}$ represents the residual defined by

$$\tilde{r}(\widetilde{W}) = \tilde{b} - \exp\left(-A\left(\frac{\widetilde{W}D}{\widetilde{W}R}\right)\right)s, \tag{2.18}$$

using as norm the Euclidean norm what we want to solve is the least squares problem

$$\min_{\widetilde{W}} \frac{1}{2}||\tilde{r}(\widetilde{W})||_2^2 \tag{2.19}$$

under the condition $\widetilde{W}\mathbf{1}_{N_m} = \mathbf{1}_{N_v}$, where with $\mathbf{1}_N$ we mean the vector of length N whose elements are all 1. Mainly thanks to this condition, after making a variable change $\widetilde{w}_{l,1} = 1 - \sum_{m=2}^{N_m} \widetilde{w}_{l,m}$, using the Matlab-like notation $\overline{W} = \widetilde{W}(:, 2 : N_m)$, (2.18) becomes

$$r(\overline{W}) = \tilde{b} - \exp\left(-A\left(\frac{\left[1 - \sum_{m=1}^{N_m-1} \overline{W}(:,m), \overline{W}\right] D}{\left[1 - \sum_{m=1}^{N_m-1} \overline{W}(:,m), \overline{W}\right] R}\right)\right)s. \tag{2.20}$$

with $r : \mathbb{R}^{N_v \cdot (N_m - 1)} \longrightarrow \mathbb{R}^{N_p \cdot N_\theta}$. Then the problem (2.19) comes back to find

$$\min_{\overline{W}} \frac{1}{2} ||r(\overline{W})||_2^2 = \min_{\overline{W}} \mathcal{F}(\overline{W}) \tag{2.21}$$

where $\mathcal{F}(\overline{W}) = \frac{1}{2}||r(\overline{W})||_2^2$ is the objective function of the minimum problem we want to solve.

Since $r : \mathbb{R}^{N_v \cdot (N_m - 1)} \longrightarrow \mathbb{R}^{N_p \cdot N_\theta}$, depending on the chosen parameters the problem can be:

1. **overdetermined**, if $N_v \cdot (N_m - 1) < N_p \cdot N_\theta$, that is there are more equations than unknowns;

2. **underdetermined**, if $N_v \cdot (N_m - 1) > N_p \cdot N_\theta$, that is there are more unknowns than equations.

In operational terms, the two cases cannot be resolved in the same way because the algorithms which are efficient for one case could be ineffective for the other one. In our work we will always consider the overdetermined case: in particular, if we consider for example $N_v = x_1 \cdot x_2 \cdot x_3$ and $N_p = x_1 \cdot x_2$, in order to lead back to the first case we will choose the parameters such that $x_3 \cdot (N_m - 1) < N_\theta$.

2.6 Regularization of the problem

In general non linear least-squares problem in (2.21) turns out to be ill-conditioned, since ill-conditioned turns out to be the Jacobian matrix of residuals vector $r(\overline{W})$. Therefore, in order to obtain a good solution, the method needs to be regularized.

A first type of regularization that we study in this thesis is the application

of the Levenberg-Marquardt method that we presented in Chapter 1, namely an iterative method of the form

$$\overline{W}_{k+1} = \overline{W}_k + \alpha_k p_k \tag{2.22}$$

where the direction p_k is calculated by solving the linear system

$$(J_k^T J_k + \lambda_k I)p_k = -J_k^T r_k \tag{2.23}$$

and the step length α_k is determined by using the Armijo's formula with backtracking.

This method differs from Gauss-Newton method for the addition of the term $\lambda_k I$ in equation (2.23), and it is exactly the presence of that additional term which makes the Levenberg-Marquardt method a regularization method. In fact, since the matrix $J_k^T J_k$ turns out to be ill-conditioned, adding to it a diagonal matrix composed of positive elements ensures that the conitioning number decreases and permits to obtain a better solution of the linear system (2.23).

So, the problem will be how to chose the λ_k numbers; as we will explain later, in our experiments we tested some available choices of λ_k. A first strategy will be to choose

$$\lambda_k = c$$

constant at each step k; another one will be to choose

$$\lambda_0 = c_2, \quad \lambda_k = \max\left\{c_1, \min\left\{c_2, ||J_k^T r_k||\right\}\right\}$$

with $0 < c_1 < \lambda_k < c_2$, that ensures to accelerate the convergence for small residuals, or to choose

$$\lambda_k = ||J_k||^2$$

where the norm can be the 2-norm or the Frobenius norm.

The real resolution of the system (2.23) has been realized by implementing the iterative conjugate gradient method and the global convergence of the Levenberg-Marquardt method is assured by the line search strategy given by (2.22).

2.7 Gradient vector approximation

In this paragraph we want to present an expression for the gradient of the objective function

$$\mathcal{F}(\overline{W}) = \frac{1}{2}||r(\overline{W})||_2^2 \tag{2.24}$$

since it is indispensable in order to resolve the minimum problem we are studying.
As we said in Chapter 1, in the new notation the gradient of (2.24) takes the form

$$\nabla\mathcal{F}(\overline{W}) = J(\overline{W})^T r(\overline{W})$$

where

$$J(\overline{W}) = \left[\frac{\partial r_k}{\partial w_{i,j}}\right] = \begin{bmatrix} \nabla r_1(\overline{W})^T \\ \nabla r_2(\overline{W})^T \\ \vdots \\ \nabla r_{N_p \cdot N_\theta}(\overline{W})^T \end{bmatrix}, \quad \begin{array}{l} k = 1, 2, \ldots, N_p \cdot N_\theta \\ i = 1, 2, \ldots, N_v \\ j = 1, 2, \ldots, N_m - 1. \end{array} \tag{2.25}$$

Since $r(\overline{W})$ is given by (2.20), we still have to find an expression for $J(\overline{W})$. Instead of calculating exactly its expression, we decided to approximate it with a finite difference approximation, in particular with the first order centred finite difference that we presented in Section 1.4. More precisely, this technique can be used to approximate the full Jacobian $J(\overline{W})$ by applying the (1.30) to (2.25) one column at a time and taking as ε the square root of the machine precision.

Chapter 3

Numerical results

In this chapter we are going to present and comment the results of the experiments for the breast imaging reconstruction described by the model shown in Section 2.5 using the numerical methods introduced in Chapter 1. The first set of methods is linked to the Levenberg-Marquardt method presented in Section 1.3.3. As we said in (1.20), the Levenberg-Marquardt method is an iterative method of the form

$$x_{k+1} = x_k + \alpha_k p_k \tag{1.3}$$

where the direction p_k is given by the solution of the system

$$(J_k^T J_k + \mu_k I)p_k = -J_k^T r_k \tag{1.20}$$

and μ_k are some nonnegative numbers. In this chapter we tested different possible choices for the values of μ_k and we will present in the following sections all the relative results.

As regards the computation of the step length α_k, in all our tests we computed it with the backtracking method described by Algorithm 1.

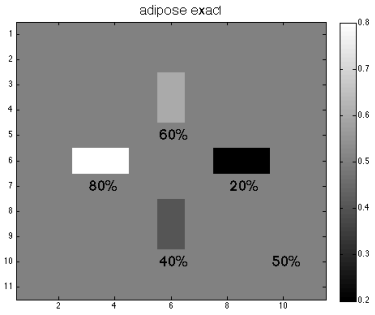
Since the system (1.20) can be solved directly or by using an iterative method, from now on we use the adjective *external* with regard to the Levenberg-Marquardt method, while we use the adjective *internal* with regard to the computation of p_k in the linear system (1.20).

The second set of methods instead belongs to the L-BFGS class, namely those algorithms which implement the BFGS formulas using a limited quantity of storage. We will test two methods, namely those ones described by Algorithm 6 and Algorithm 8 presented in Section 1.3.4. Before starting with the description of the numerical results, we illustrate all the parameters that have been chosen to create the test problem.

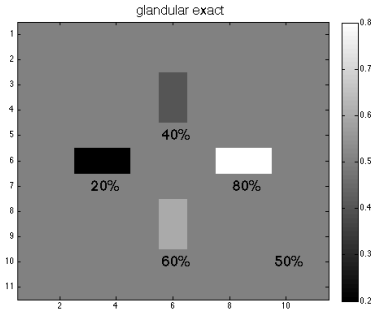
3.1 Test problem

In our test problem for a simulated breast imaging reconstruction we considered one simulated three-dimensional phantom object of size $11 \times 11 \times 5$ made of four ellipses consisting of a tissue mixture with varying percentages of glandular and adipose tissue, while the background is made of a mixture of 50% adipose and 50% glandular tissue. In Figure 3.1 we can see the central slice of the exact object with the percentage of the mass attenuation coefficients of the two tissues. Since we tested a quite small problem to limit the computational time, our ellipses are in this case rectangles; if we choose a high dimensional test problem, these sections would be similar to ellipses. We can note that the mass attenuation coefficients of the glandular material percentages sum together with the adipose material coefficients to 1, because for the (2.10) we have that

$$\sum_{m=1}^{N_m} \widetilde{w}_{l,m} = 1, \quad l = 1, 2, \ldots, N_v.$$



(a) Exact adipose tissue.



(b) Exact glandular tissue.

Figure 3.1: Central slice of the exact three-dimensional phantom object with the percentages of different tissues.

As we saw in Chapter 2, the distributions of the mass coefficients weigths $\widetilde{w}_{l,m}$ are stored in the matrix $\overline{W}_{ex}$ of dimensions $N_v \times N_{m-1}$; so, in this case, since we have chosen $N_v = 11 \times 11 \times 5$ and $N_m = 2$, $\overline{W}_{ex}$ is a column vector of dimensions 605×1.

As regards the construction of the ray trace matrix A, it has been obtained by using the fast algorithm for the calculus of an exact radiological path for a three-dimensional CT system presented by Robert L. Siddon in [5]. More precisely, in this work the matrix A has been constructed by using the Siddon's algorithm with $N_\theta = 11$ equispaced projection angles from $-17°$ to $17°$ and $N_\varepsilon = 37$ different levels of energy from 10 keV to 28 keV in 0.5 keV steps.

As we told in Chapter 2, what we expect to find is a weight combination $\widetilde{W}$ that, from all the available ones, is such that the distance between observed and predicted is small, namely we try to solve

$$\min_{\overline{W}} = \frac{1}{2} \left\| b - \exp\left(-A \left(\frac{\widetilde{W} D}{\widetilde{W} R} \right) \right) s \right\|_2^2$$

where

$$b = \exp\left(-A\left(\frac{\widetilde{W}_{ex}D}{\widetilde{W}_{ex}R}\right)\right)s + \eta$$

are observed data corrupted by the noise η. The type of noise we consider in our experiments is a Gaussian noise, and when we talk about the noise level nl we mean the relative noise level, that is the ratio between $\|\eta\|$ and $\|b - \eta\|$:

$$nl = \frac{\|\eta\|}{\|b - \eta\|}$$

and, from now on, when we talk about the errors we always refer to a relative error, that is

$$err = \frac{\|\overline{W}_{ex} - \overline{W}\|}{\|\overline{W}_{ex}\|}.$$

In all the tests, when we used an iterative method described in Chapter 1 we started as initial iterate with the column vector made of all 0.5. Finally, since $x_3 = 5$, $N_m = 2$ and $N_\theta = 25$, as we said at the end of the Section 2.5, the relation

$$x_3 \cdot (N_m - 1) < N_\theta$$

is satisfied and so we have to solve an overdetermined problem.

After that we have made some considerations about this small problem, at a later stage we will make some tests for a bigger one, more precisely of size $31 \times 31 \times 7$.

3.2 Numerical results for the Levenberg-Marquardt method

3.2.1 The Levenberg-Marquardt method with constant μ_k

In this first experiments set we used the Levenberg-Marquardt method (1.20) where the coefficients $\mu_k = \mu$ constant at every iteration k while we solved the system iteratively by using the conjugate gradient method given by Algorithm 2.
We have chosen to consider 30 values for μ from 10^{-5} to 10^{10} uniformly in logarithmic scale and to use this stop condition:

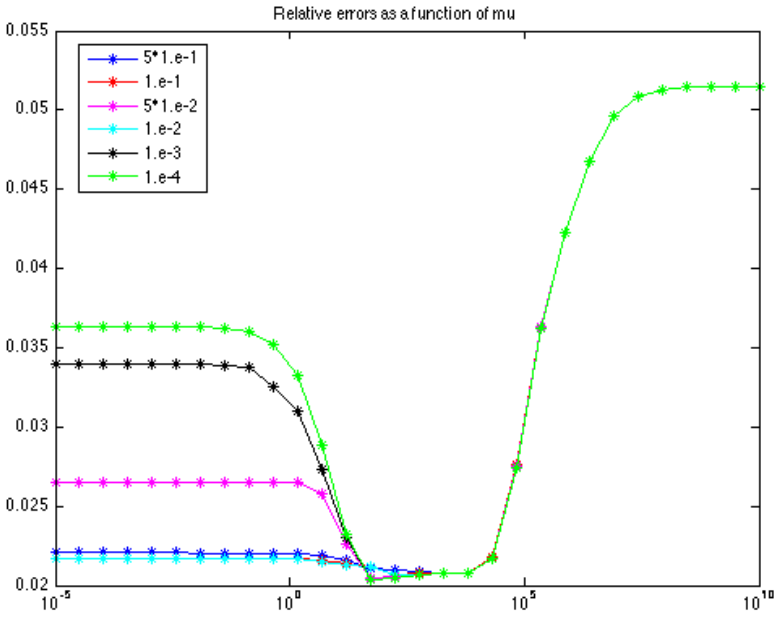
$$\textbf{while } (k \leq 100) \ \& \ (err_k \leq err_{k-1}) \ \& \ \left(err_{k-1} - err_k \geq 10^{-5}\right) \textbf{ do}, \quad (3.1)$$

that is, the Levenberg-Marquardt method ends when either the maximum number of iterations in reached, either we reach the semi-convergence or the error becomes too flat.
For the solution of the (1.20) we have chosen this stop condition:

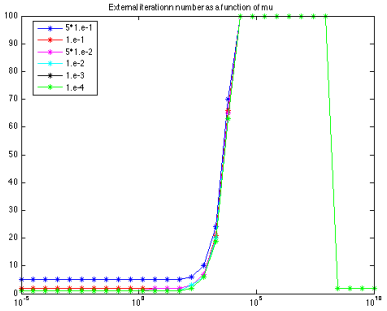
$$\textbf{while } (j \leq 500) \ \& \ (\|r_j\| > tol \cdot \|r_0\|) \textbf{ do}, \quad (3.2)$$

that is, the CG method ends when either the maximum number of internal iterations is reached or the relative residual becomes smaller than the internal tolerance *tol*. In the Figures 3.2 and 3.3 we can see several curves: each one is relative to a different value for the tolerance *tol* from the set

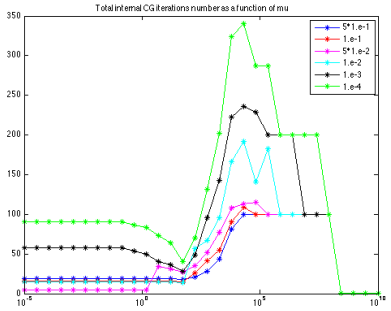
$$tol = \left[5 \cdot 10^{-1}, \ \ 10^{-1}, \ \ 5 \cdot 10^{-2}, \ \ 10^{-2}, \ \ 10^{-3}, \ \ 10^{-4}\right].$$



(a) Plot of the relative errors as a function of the μ values.

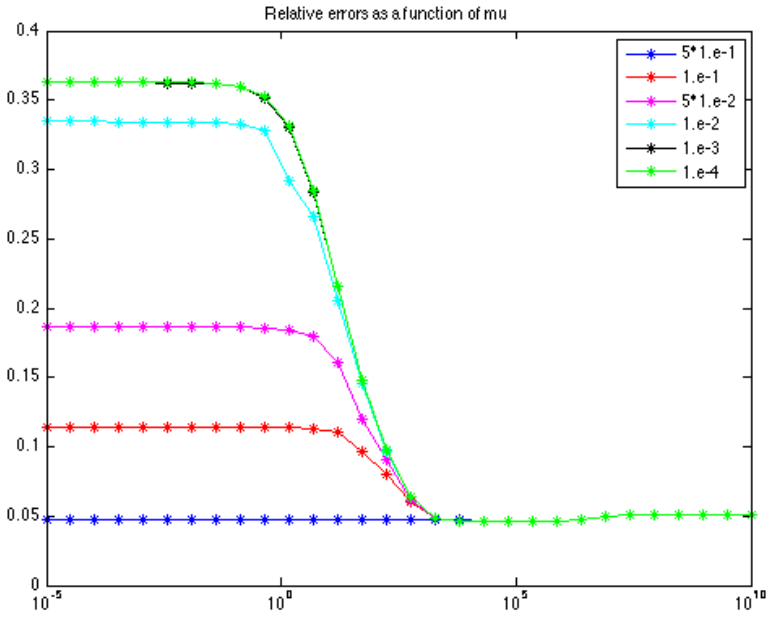


(b) Plot of the external iterations number as a function of the μ values.

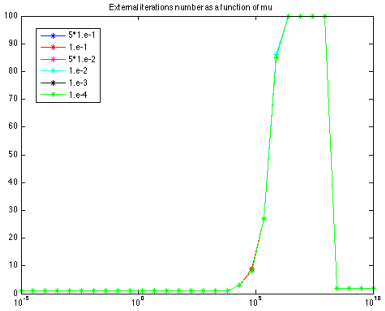


(c) Plot of the internal CG iterations number as a function of the μ values.

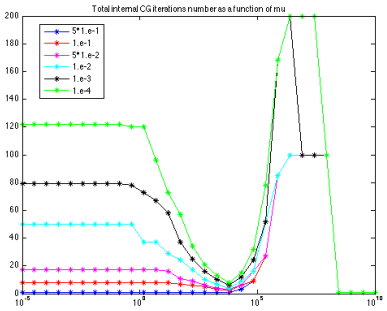
Figure 3.2: Results for the Levenberg-Marquardt method with μ constant with noise level $nl = 10^{-3}$



(a) Plot of the relative errors as a function of the μ values.



(b) Plot of the external iterations number as a function of the μ values.



(c) Plot of the internal CG iterations number as a function of the μ values.

Figure 3.3: Results for the Levenberg-Marquardt method with μ constant with noise level $nl = 10^{-2}$

We plotted three different quantities: the relative error of the external iterations, the number of the external iterations and the number of the internal CG iterations calculated to reach it. Each of these quantities is as a function of the value of μ and, as we can see from the legend, each colour refers to a different internal tolerance *tol*.
In Figure 3.2 we can see the results for a noise level $nl = 10^{-3}$, in Figure 3.3 for a noise level $nl = 10^{-2}$.

From these figures we note that for high values of μ the behaviour of the method is similar for all the tolerances, even if (as we expected) the number of internal iterations is higher for lower tolerances. Moreover, for high values of μ the error curve starts to raise (more for $nl = 10^{-3}$, slightly for $nl = 10^{-2}$) and the method generally ends for the condition on the flat error.

In the central part of the figure we have the minima of the error, while for the smallest value of μ the curves go stabilizing to a certain constant value depending on the tolerance.

In Table 3.1 we present for different tolerances *tol* of the CG stopping condition (3.2) the value of μ that minimizes the relative error (respectively μ_{best} and err_{best}), together with the number of external and internal iterations (k_{best} and $itCG_{best}$) relative to $nl = 10^{-3}$. We note that the value of the minimal errors are not very different from each other, while the number of iterations varies significantly.

3.2.2 $\mu_k \equiv 0$: Gauss-Newton method

Looking at the Figure 3.2 and Figure 3.3 we can see that for small values of $\mu_k = \mu$ we have a constant behaviour that, is some cases, corresponds to a small relative error value. So, starting from this idea, we tried to test the

Table 3.1: Best values for Levenberg-Marquardt method with μ constant with noise level $nl = 10^{-3}$ for different values of tol.

tol	μ_{best}	err_{best}	k_{best}	$itCG_{best}$
$5 \cdot 10^{-1}$	$6.210169e+03$	$2.077937e-02$	70	81
10^{-1}	$1.743329e+02$	$2.069595e-02$	3	26
$5 \cdot 10^{-2}$	$5.298317e+01$	$2.050922e-02$	2	27
10^{-2}	$1.743329e+02$	$2.065558e-02$	3	57
10^{-3}	$5.298317e+01$	$2.035240e-02$	1	28
10^{-4}	$5.298317e+01$	$2.032629e-02$	1	40

choice of

$$\mu_k = \mu = 0$$

at each iteration. This choice, as we observed in the Section 1.3.3, lead us back to the Gauss-Newton method, that is the Levenberg-Marquardt method without any regularization.

The results are in the Table 3.2, where we can see the global error (err), together with the number of internal and external iterations (respectively k and $itCG$) relative to $nl = 10^{-3}$.

As we can see by comparing the Figure 3.2 and Table 3.2, the curves probably remain constant up to $\mu = 0$. So, since in general making a good choice for μ is a difficult problem, these results suggest that we could choose $\mu = 0$ for certain values of tolerance.

Moreover, observing the relative errors for the several tolerances we note that we are not forced to use a very small internal tolerance: in this case we can obtain a good solution with a not so small tolerance, such as $tol = 10^{-1}$, and in this way we should make few internal iterations.

Table 3.2: Statistics for the Gauss-Newton method ($\mu = 0$) with noise level $nl = 10^{-3}$

tol	err	k	$itCG$
$5 \cdot 10^{-1}$	$2.204699e-02$	5	19
10^{-1}	$2.170383e-02$	2	16
$5 \cdot 10^{-2}$	$2.648451e-02$	1	5
10^{-2}	$2.169613e-02$	1	15
10^{-3}	$3.398179e-02$	1	58
10^{-4}	$3.632318e-02$	1	91

Now we solve the previous Gauss-Newton method

$$J_k^T J_k p_k = -J_k^T r_k \tag{1.20}$$

by using another stop condition on the computation of the direction p_k, namely a condition suggested by the CGLS method presented in Section 1.2.4

Fixed $0 < \rho < 1$, at every external iteration k the new stop condition requires to take as p_k the minimum value such that

$$\|J_k p_k + r_k\| \geq \rho \|r_k\|.$$

So, using the CGLS notation, this rule corresponds to use Algorithm 3 with the following stop condition

$$\textbf{while } (k \leq 500) \ \& \ (\|J_k p_j + r_k\| > \rho \|r_k\|) \textbf{ do} \tag{3.3}$$

and taking as p_k not the last iteration, but the penultimate one.

First of all we tested this method for 100 values of $\rho \in (0, 1)$ and with a noise level $nl = 10^{-2}$. The relative errors and the internal CG iterations number as a function of ρ are in Figure 3.4, while for all ρ the number of external iterations is equal to 1. As we can see, this method results more

efficient for values of ρ near to 1, because the errors are low and a few iterations are necessary. Although, the error values are higher than the other methods seen before: for example, for $\rho = 0.9$ we have $err = 5.142595e-02$.

If we make the same test for a noise level $nl = 10^{-3}$ (Figure 3.9), we note that for values of ρ near to 1 the error keeps on being higher than the previous methods and that an accurate choice of ρ in order to get the minimum error value is difficult to make. We can conclude that the stopping rule (3.3) gives worse results than stopping rule (3.1). Hence, in the following tests we will always use the criterion (3.3).

3.2.3 The Levenberg-Marquardt method with varying μ_k

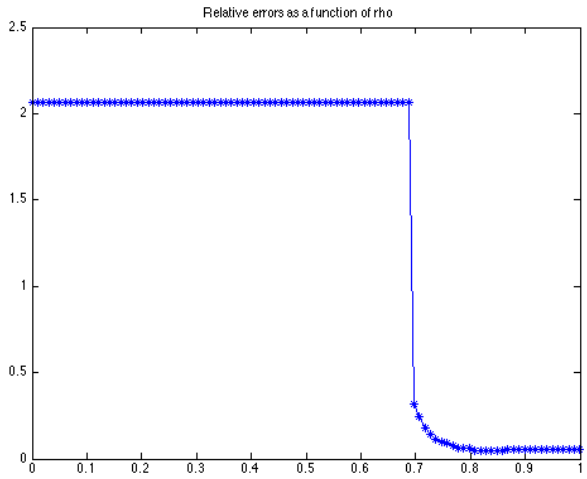
Now we are going to present the numerical results for a Levenberg-Marquardt method where the coefficients μ_k vary at each external iteration. The value of μ_k is defined at each iteration as

$$\mu_k = \max\left\{c_1,\ \min\left\{c_2, \|J_k^T r_k\|\right\}\right\}, \quad \mu_0 = c_2 \tag{3.4}$$

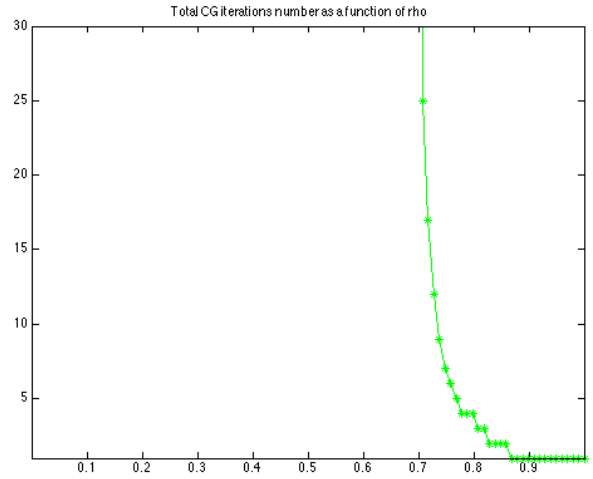
with $0 < c_1 < \mu_k < c_2$. This strategy, suggested in [9], generates a decreasing sequence of μ_k and starts from the idea that the convergence is accelerated when the residuals are small. The internal system has been solved by using the conjugate gradient method with different tolerances.

In our experiments we set the constants as $c_1 = 10^{-2}$, $c_2 = 10^2$ and considered every time a different internal tolerance from the vector

$$tol = \left[5 \cdot 10^{-1},\ \ 10^{-1},\ \ 5 \cdot 10^{-2},\ \ 10^{-4}\right].$$

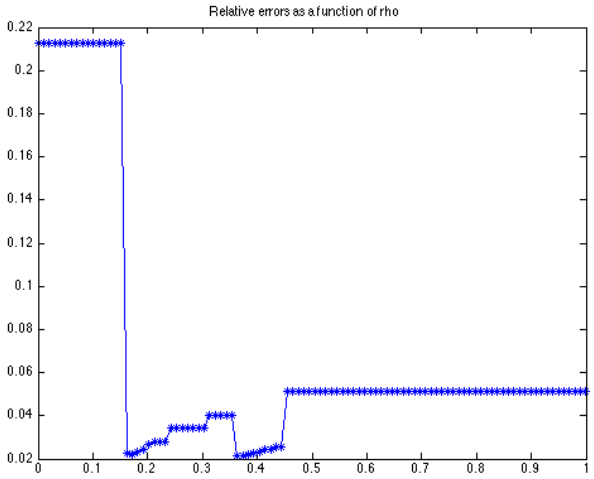


(a) Plot of the relative errors as a function of the ρ values.

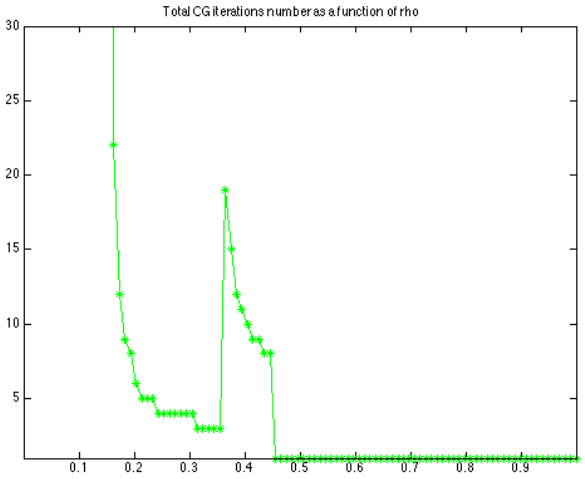


(b) Plot of the internal iterations number as a function of the ρ values.

Figure 3.4: Results for the Gauss-Newton method with the stop condition given by (3.3) and noise level $nl = 10^{-2}$

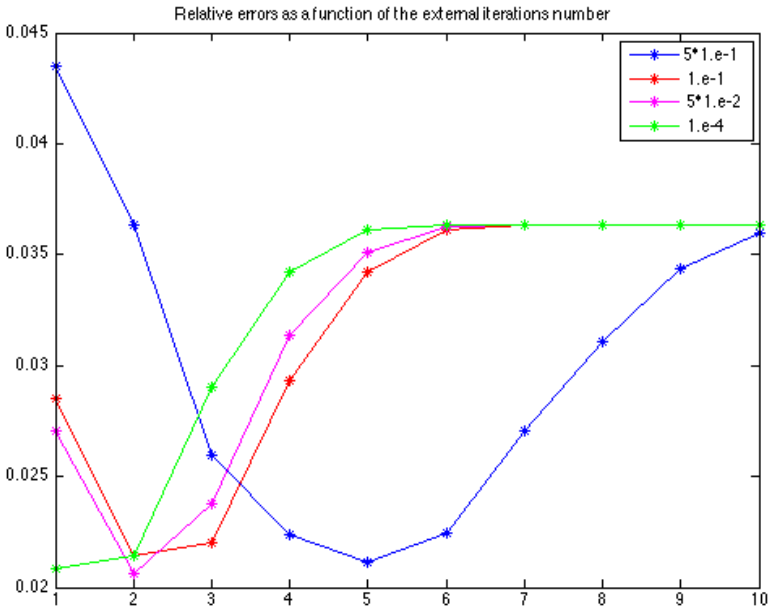


(a) Plot of the relative errors as a function of the ρ values.

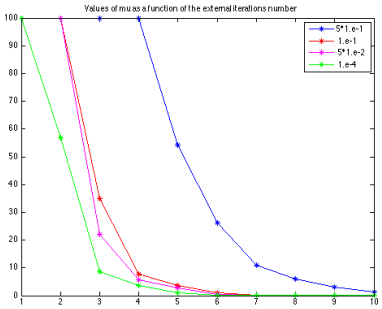


(b) Plot of the internal iterations number as a function of the ρ values.

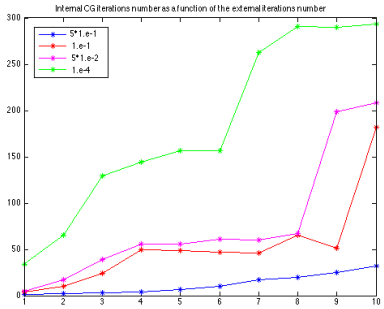
Figure 3.5: Results for the Gauss-Newton method with the stop condition given by (3.3) and noise level $nl = 10^{-3}$



(a) Plot of the relative errors as a function of the external iterations.

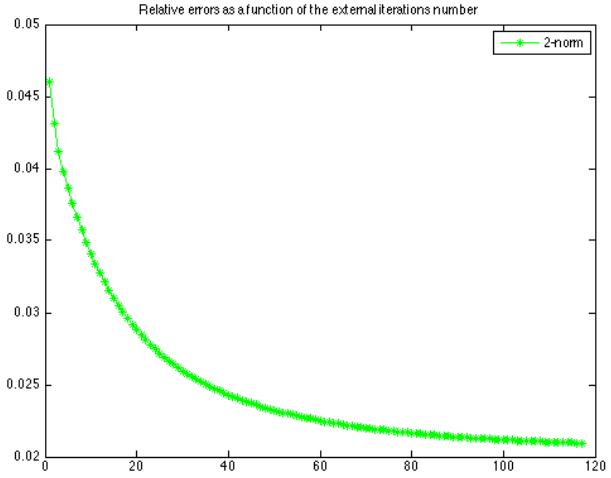


(b) Plot of the μ_k values as a function of the external iterations.

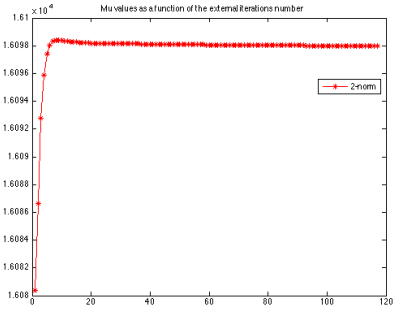


(c) Plot of the internal CG iterations versus the external iterations.

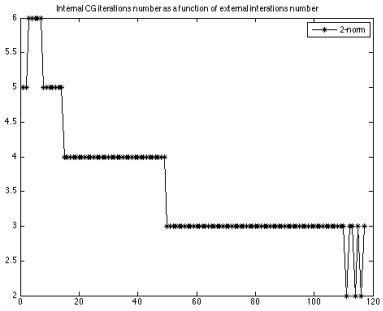
Figure 3.6: Results for the Levenberg-Marquardt method with varying μ_k as in (3.4) with noise level $nl = 10^{-3}$



(a) Plot of the relative errors as a function of the external iterations.



(b) Plot of the μ_k values as a function of the external iterations.



(c) Plot of the internal CG iterations versus the external iterations.

Figure 3.7: Results for the Levenberg-Marquardt method with varying μ_k as in (3.5) with noise level $nl = 10^{-3}$

In Figure 3.6 there are several curves relative to a noise level $nl = 10^{-3}$ and each of them, as we read from the legend, refers to a different internal tolerance value. In particular, in Figure 3.6(a) we can see the error as a function of the external iteration and the semi-convergence behaviour, in Figure 3.6(b) the decreasing sequence of μ_k and in Figure 3.6(c) the total number of the internal CG iterations.
From these figures we note that when the internal tolerances increases, the the number of external iterations corresponding to the minimum of the error increases, while the number of the internal iterations decreases, consistently. We note also that by taking a not so small internal tolerance value, making a few external iterations and a limited number of internal iteration we could get a relative error value that is comparable with what we obtained by choosing a constant μ or $\mu = 0$.

Now we are going to consider a new way of choosing μ_k at each iteration. So, instead of using (3.4), we set the value of μ_k as

$$\mu_k = \|J_k\|^2 \tag{3.5}$$

namely the square of the norm of the Jacobian, computed using some kind of norm.

In Figure 3.7 we can see some graphs relative to a noise level $nl = 10^{-3}$ and the choice of using the 2-norm in order to calculate (3.5). In particular, in Figure 3.7(a) there is the relative error as a function of the external iterations number. We note that many iterations are necessary and we do not reach the semi-convergence behaviour, since the methods end for the condition on the flat error. In Figure 3.7(b) the sequence of μ_k as a function of the external iterations is plotted. They seem to be quite constant, so it seems unnecessary to calculate μ_k with (3.5) at each iteration when we could choose a μ constant

for all iterations as we did in the previous methods. Moreover, comparing Figure 3.7(a) with Figure 3.2(a), we can note that the error value at the exit in Figure 3.7(a) is quite high compared to what we could obtain by taking μ constant in the same order as that achieved by Figure 3.7(b). So, because of its slowness, this choice of μ_k seems to be inefficient.

3.2.4 Comparisons between methods

In this section we are going to compare the several methods we analysed and highlight pros and cons.

From what we said in the previous sections, the Gauss-Newton method with the stop condition (3.3) and the Levenberg-Marquardt method with varying μ_k as in (3.5) seem to be inefficient. In fact, the former do not give a clear criterion about the best choice of ρ and, for ρ near to 1, the error value is still higher than other methods. Instead for the latter too many iterations are necessary to reach the stop condition and we need to calculate at every iteration the norm of the Jacobian. Later we found that the μ_k are quite constant, so we could use a μ constant in order to save computational time by not calculating the norm of the gradient and making less iterations.

For a noise level $nl = 10^{-3}$, taking into account the relative errors and the number of internal and external iterations, the best methods that we analysed seem to be the Levenberg-Marquardt method with μ constant in

Table 3.3: Statistics of the Levenberg-Marquardt method for the test problem of size $11 \times 11 \times 5$.

Method	err	k	$itCG$	$time$
$\mu = 10^2$	$2.111874e-02$	5	16	16.0
$\mu = 0$	$2.204699e-02$	5	19	19.0
$\mu_k = \max\left\{10^{-2},\ \min\left\{10^2, \|J_k^T r_k\|\right\}\right\}$	$2.116727e-02$	5	17	17.0

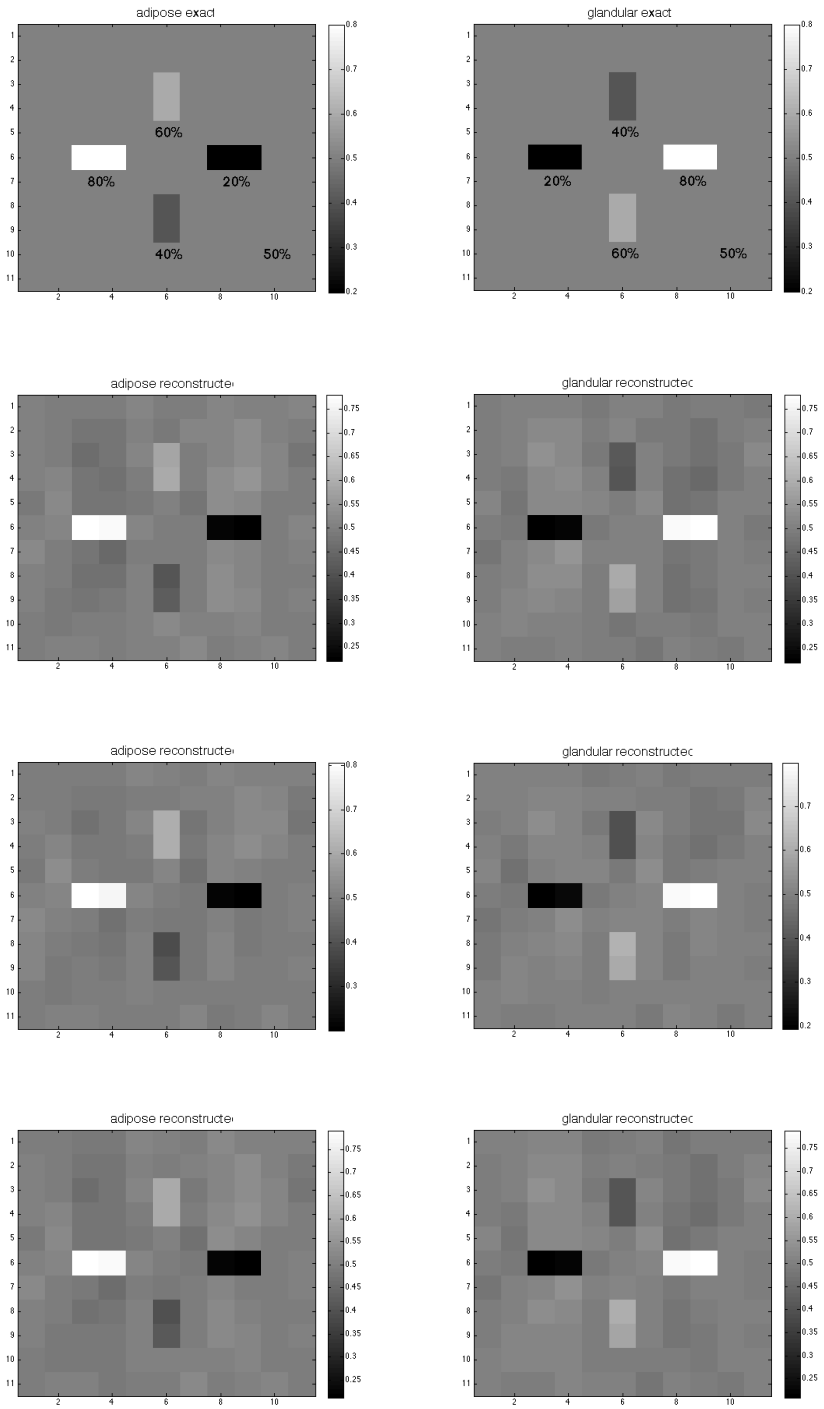


Figure 3.8: Reconstructed images for the test problem of size $11 \times 11 \times 5$. 1st row: exact images. 2nd row: reconstructed images for Levenberg-Marquardt method with $\mu = 10^2$. 3rd row: reconstructed images for Gauss-Newton method ($\mu = 0$). 4th row: reconstructed images for Levenberg-Marquardt method with μ_k varying as (3.4).

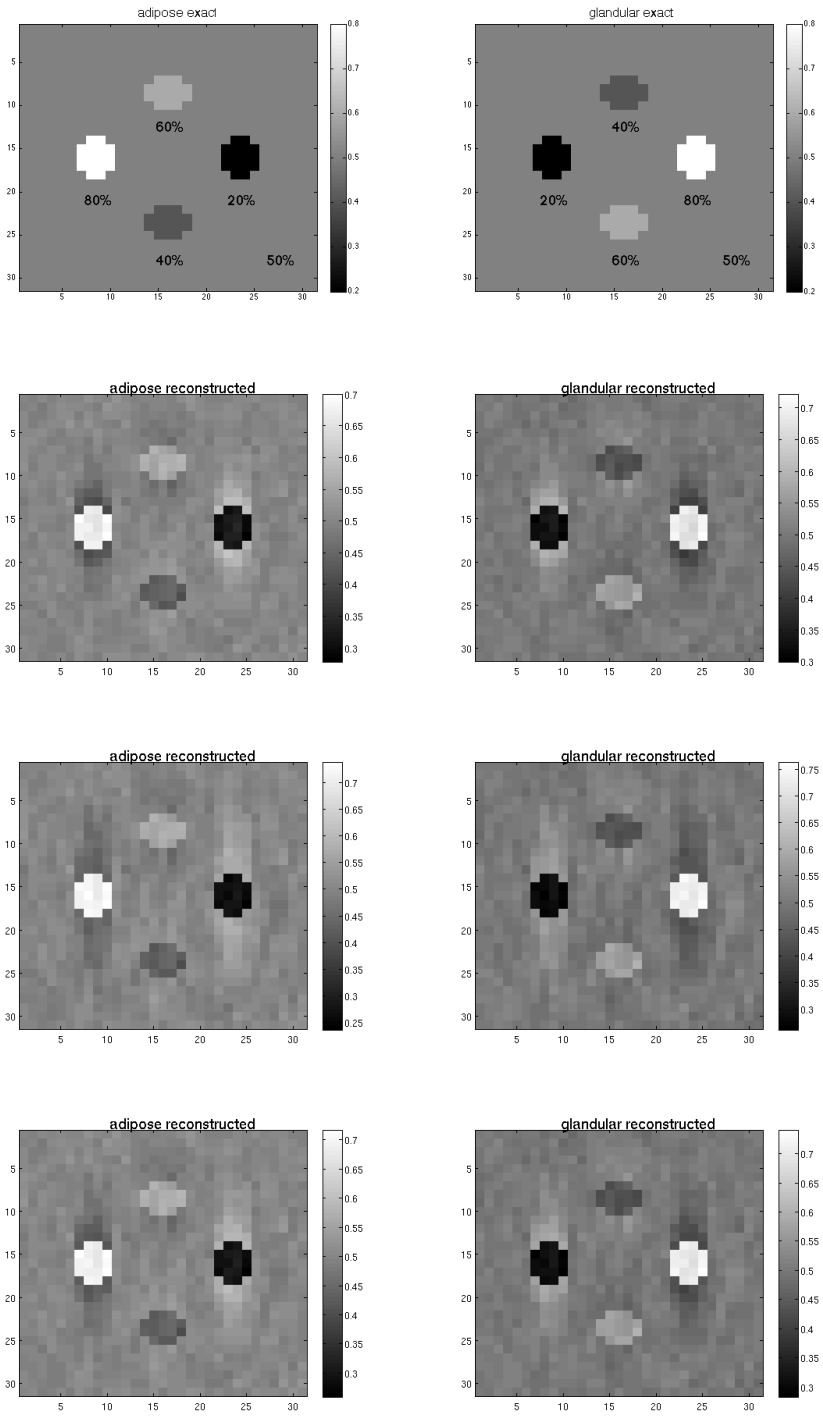


Figure 3.9: Reconstructed images for the Levenberg-Marquardt method and the test problem of size $31 \times 31 \times 7$.
1st row: exact images. 2nd row: reconstructed images for Levenberg-Marquardt method with $\mu = 10^2$. 3rd row: reconstructed images for Gauss-Newton method ($\mu = 0$). 4th row: reconstructed images for Levenberg-Marquardt method with μ_k varying as (3.4).

the order of 10^2, the Gauss-Newton method ($\mu = 0$) and the Levenberg-Marquardt method with μ_k varying as

$$\mu_k = \max\Big\{c_1,\ \min\big\{c_2, \|J_k^T r_k\|\big\}\Big\}, \quad \mu_0 = c_2 \tag{3.4}$$
$$c_1 = 10^{-2}, \quad c_2 = 10^2$$

all these with an internal CG tolerance value not necessary so small. In the Table 3.3 we sum up the statistics of these three methods for an internal CG tolerance $tol = 5 \cdot 10^{-1}$ and $nl = 10^{-3}$: we can read the relative errors, the internal and external iterations number for $\mu = 10^2$, $\mu = 0$ and μ_k as in (3.4).

As we can see from Table 3.3, all these method are comparable, since the relative errors and the iterations number are roughly the same. Only for the Gauss-Newton method the performance is slightly less efficient, but it has the advantage that the user has not to make any choice of the parameters, such as, for example, the constant value μ in the first method or the constants c_1 and c_2 in the third one.

Now let us have a look on the reconstructed images for these three methods. In Figure 3.8 we can see the central slice of the reconstructed object, where in the left column is plotted the adipose tissue percentage and in the right column the glandular tissue one. From the reconstruction we find again that the three method are comparable, since in the three cases the reconstructed images have all roughly the same quality.

Table 3.4: Statistics of the Levenberg-Marquardt method for the test problem of size $31 \times 31 \times 7$.

Method	err	k	$itCG$	$time$
$\mu = 10^2$	$3.091459e-02$	8	36	1607
$\mu = 0$	$3.640818e-02$	4	22	828
$\mu_k = \max\big\{10^{-2},\ \min\big\{10^2, \|J_k^T r_k\|\big\}\big\}$	$3.214635e-02$	7	31	1346

In conclusion, we made the same tests as above for a bigger test problem, namely we considered a three-dimensional phantom object of size $31 \times 31 \times 7$. From the Table 3.4 we can note that the semi-convergence is reached by the Gauss-Newton method ($\mu = 0$) in a lower number of internal and external iterations, but this corresponds to a higher relative error value. On the other hand, with $\mu = 10^2$ more iterations are necessary, but we have the least relative error value. However, if we look at the reconstructed images in Figure 3.9, we can observe that the quality of the results are roughly the same. We conclude that in this case, between the methods analysed, the best choice is the Gauss-Newton method, since it needs a lower iterations number for the same reconstructed image quality.

3.3 Numerical results for L-BFGS methods

In this section we are going to test the two Limited-memory BFGS algorithms that we presented in Section 1.3.4, namely Algorithm 6 and Algorithm 8. We will test some possible choices for the μ_k values starting from the smaller test problem $11 \times 11 \times 5$, then we will make some experiments for the bigger problem $31 \times 31 \times 7$ and we will compare the qualities of the reconstructed images too.

We point out that throughout the following tests we have taken a noise level $nl = 10^{-3}$, an (external) tolerance value $tol = 10^{-4}$ and $m = 5$.

3.3.1 Numerical results for Algorithm 6

We start to test Algorithm 6 by taking a varying μ_k as

$$\mu_k = \max\Big\{c_1,\ \min\big\{c_2, \|J_k^T r_k\|\big\}\Big\}, \quad \mu_0 = c_2 \tag{3.4}$$
$$c_1 = 10^{-2}, \quad c_2 = 10^2$$

with the stop condition given by

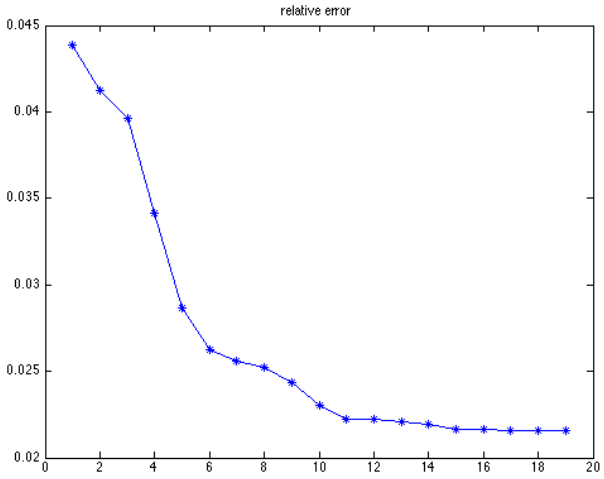
$$\textbf{while}\ \ (k \leq 100)\ \ \&\ \ (err_k \leq err_{k-1})\ \ \textbf{do} \tag{3.6}$$

In Figure 3.10a we can see the relative error as a function of the iterations number while in Figure 3.10b the gradient norm and the μ_k values at each iteration are plotted. Then we have compared the choice of taking the μ_k values varying as in (3.4) to the choice of taking μ_k constant at each iteration $\mu_k = \mu = 10^2$ with the same stop condition (3.6). In Figure 3.11 we can see the relative error as a function of the iterations number while in Table 3.5 there are the relative errors, iterations number and times for these two different choices of μ_k.

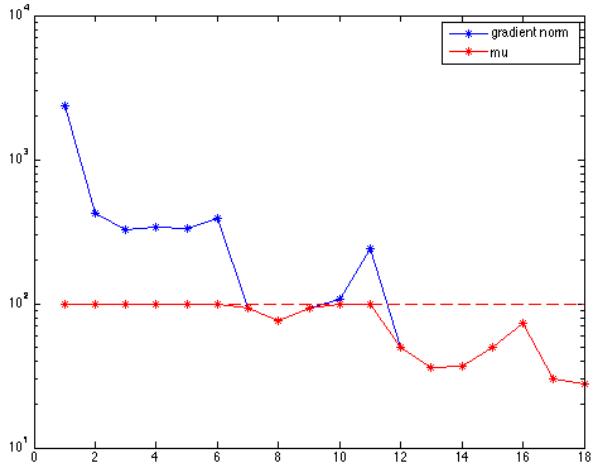
Table 3.5: Statistics of Algorithm 6 for the test problem of size $11 \times 11 \times 5$.

	err	k	$time$
$\mu_k = \max\big\{10^{-2},\ \min\big\{10^2, \|J_k^T r_k\|\big\}\big\}$	$2.153760e-02$	18	24.7
$\mu = 10^2$	$2.145430e-02$	18	26.0

Comparing the data in Table 3.5 we deduce that probably it is not necessary to take a varying μ_k value but we should investigate about which is a good constant μ value. For this goal, in Figure 3.12a, 3.12b and 3.12c respectively the relative error and the iterations number as a function of different constant μ values in $\mu = logspace(0, 4, 9)$ are plotted.



(a) Plot of the relative error versus the iterations number.



(b) Plot of the gradient norm values and the μ_k values versus the iterations number.

Figure 3.10: Results for Algorithm 6 with μ_k varying as in (3.4)

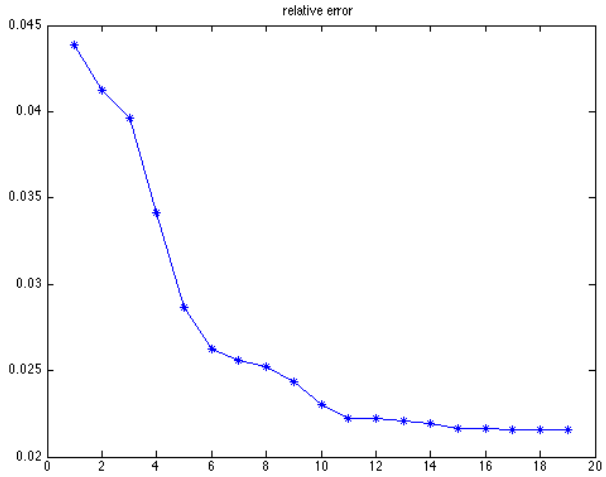


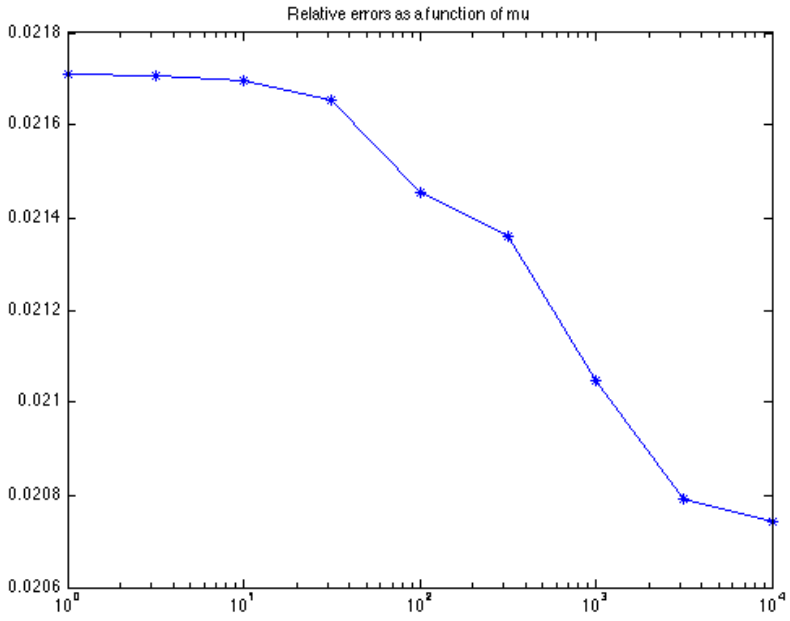
Figure 3.11: Plot of the relative error versus the iterations number for Algorithm 6 with constant $\mu_k = \mu = 10^2$.

Since for $\mu = 10^4$ we reach the maximum iterations number but, as we can see from Figure 3.11 and 3.10b the error could have a constant behaviour and so we could stop the algorithm earlier, now we introduce a new stop condition that takes into account the flatness of the error:

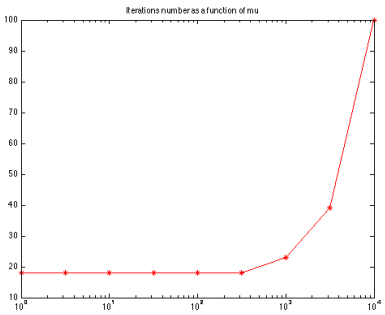
$$\textbf{while } (k \leq 100) \ \& \ (err_k \leq err_{k-1}) \ \& \ \left(err_{k-1} - err_k \geq 10^{-5}\right) \textbf{ do} \quad (3.7)$$

Now we make the same tests but with stop condition (3.7) for different constant μ values in $logspace(0, 5, 11)$. Since from Figure 3.13 we can see that for the highest values of μ the errors and the iterations number start to increase, we deduce that the errors do not have a flat behaviour but decrease slowly, and too many iterations are necessary to reach a good solution.

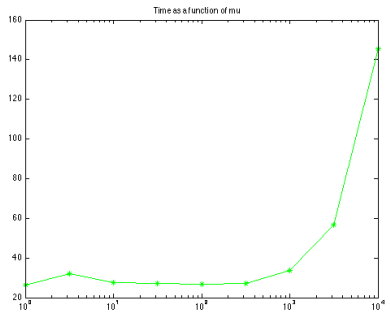
Again from Figure 3.13 we can see that, taking into account the relative error, the iterations number and the time values, a good choice of μ seems to be $\mu = 3.162 \cdot 10^2$. So, for this value, we plot in Figure 3.14 the relative error and in Table 3.6 the relative error, the iterations number and the time values.



(a) Plot of the relative errors as a function of μ values.

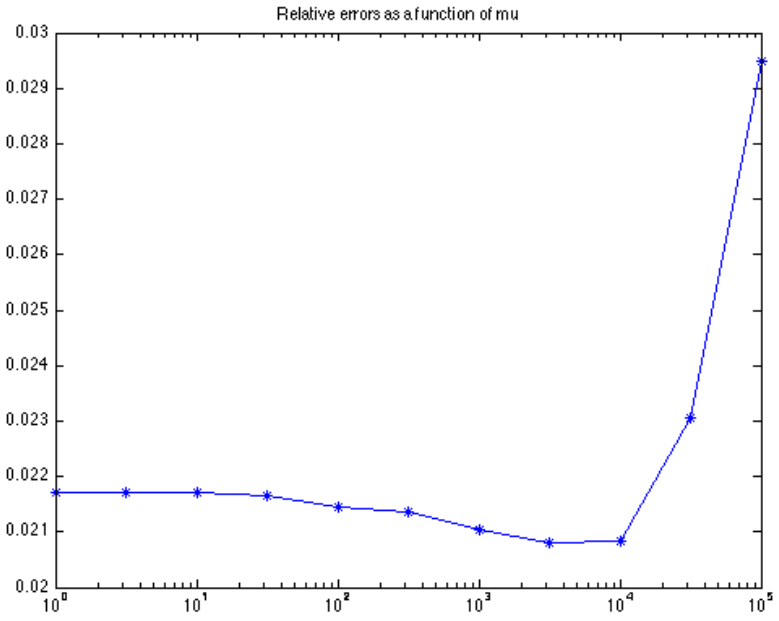


(b) Plot of the iterations number as a function of μ values.

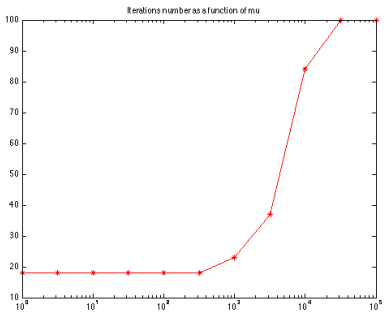


(c) Plot of the time as a function of μ values.

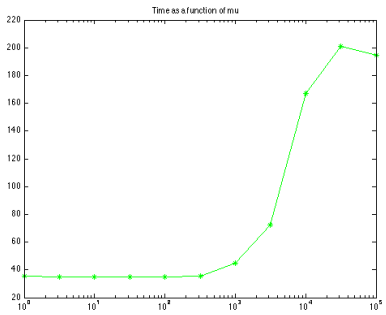
Figure 3.12: Results for Algorithm 6 for different constant μ choices in $logspace(0, 4, 9)$.



(a) Plot of the relative errors as a function of μ values.



(b) Plot of the iterations number as a function of μ values.



(c) Plot of the time as a function of μ values.

Figure 3.13: Results for Algorithm 6 for different constant μ choices in $logspace(0, 5, 11)$ and stop condition (3.7).

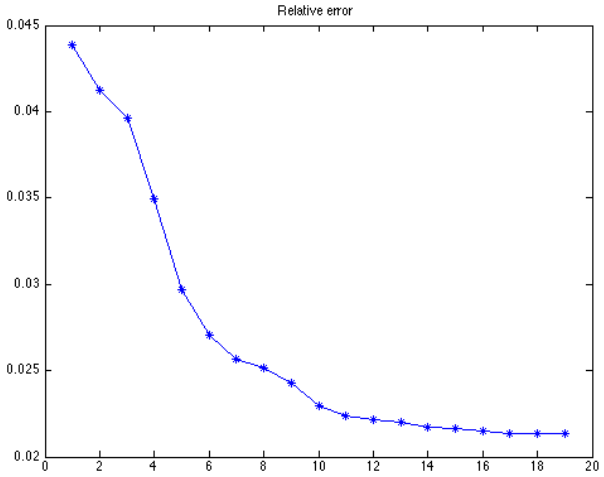


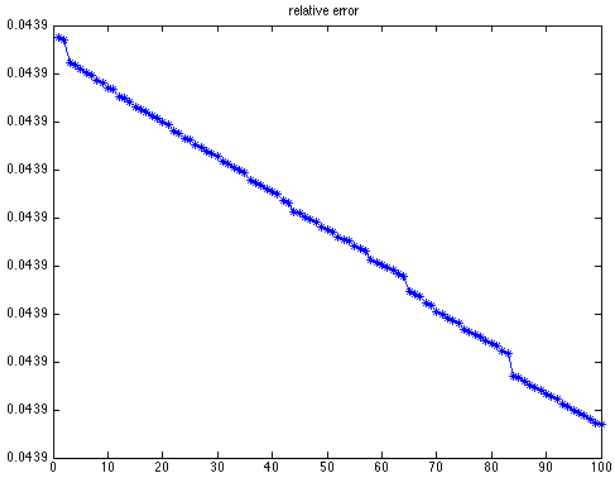
Figure 3.14: Plot of the relative error versus the iterations number for Algorithm 6 with constant $\mu_k = \mu = 3.162 \cdot 10^2$.

Table 3.6: Statistics of Algorithm 6 with $\mu = 3.162 \cdot 10^2$ for the test problem of size $11 \times 11 \times 5$.

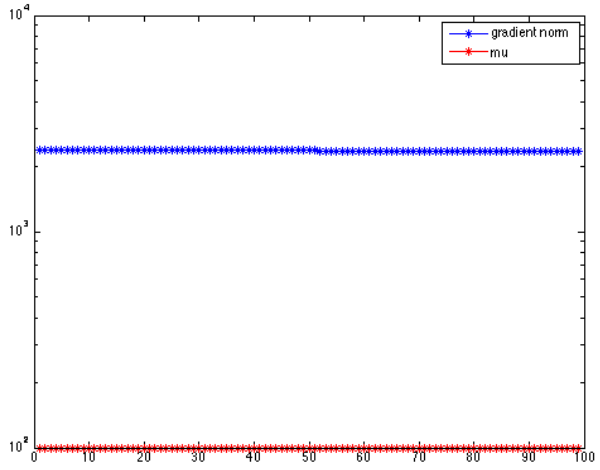
	err	k	*time*
$\mu = 3.162 \cdot 10^2$	$2.134439e-02$	18	30.0

3.3.2 Numerical results for Algorithm 8

Now we turn to consider the second L-BFGS strategy, namely that one described by Algorithm 8. In the same way as we did for Algorithm 6, we start taking a varying μ_k as in (3.4) with the stop condition given by (3.6). Since we note from Figure 3.15b that the gradient norm is quite constant, we can immediately turn to consider a $\mu_k = \mu$ constant at each iteration. More precisely, we consider directly the new stop condition given by (3.7) and we take different constant μ values in $logspace(-3, 2, 11)$. We point out that we have not considered higher values for μ because the curvature condition (1.24) has not been satisfied.

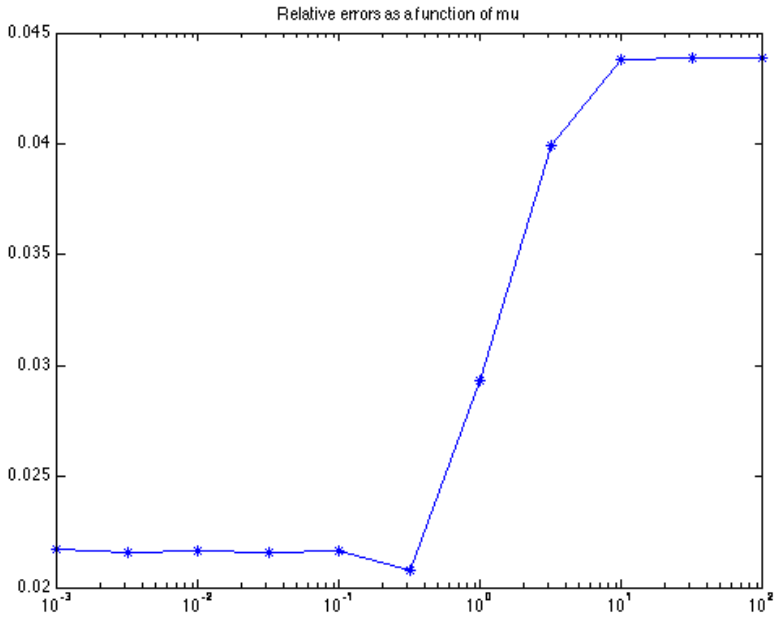


(a) Plot of the relative error versus the iterations number.

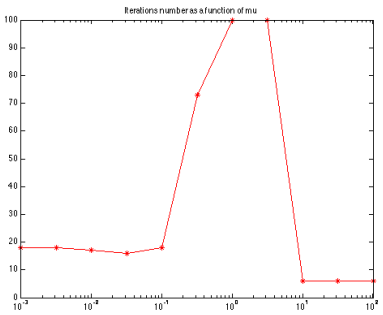


(b) Plot of the gradient norm values and the μ_k values versus the iterations number.

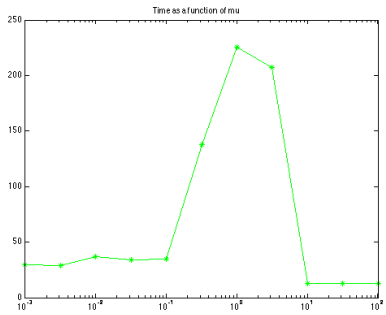
Figure 3.15: Results for Algorithm 8 with μ_k varying as in (3.4)



(a) Plot of the relative errors as a function of μ values.



(b) Plot of the iterations number as a function of μ values.



(c) Plot of the time as a function of μ values.

Figure 3.16: Results for Algorithm 8 for different constant μ choices in $logspace(-3, 2, 11)$ and stop condition (3.7).

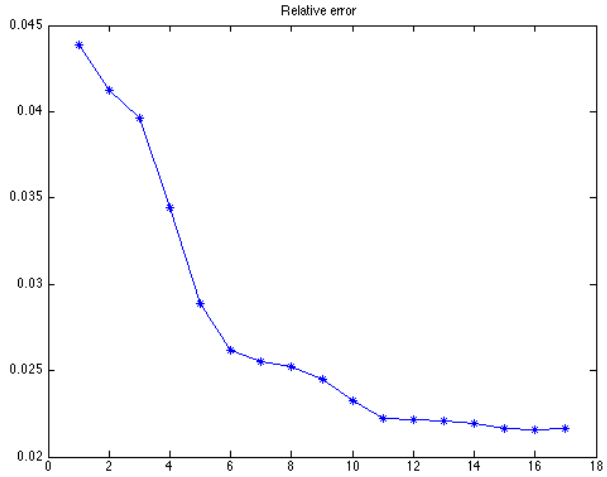


Figure 3.17: Plot of the relative error versus the iterations number for Algorithm 8 with constant $\mu_k = \mu = 3.162 \cdot 10^{-2}$.

According to Figure 3.16 we can say that, taking into account the relative error, the iterations number and the time values, $\mu = 3.162 \cdot 10^{-2}$ seems to be a good choice of μ. So, for this value, we plot in Figure 3.18 the relative error and in Table 3.7 the relative error, the iterations number and the time values.

Table 3.7: Statistics of Algorithm 8 with $\mu = 3.162 \cdot 10^{-2}$ for the test problem of size $11 \times 11 \times 5$

	err	k	$time$
$\mu = 3.162 \cdot 10^{-2}$	$2.160192e-02$	16	25.7

3.3.3 Comparisons between methods

Now let us make some remarks about the L-BFGS methods and compare the numerical results with those of the Levenberg-Marquardt method that we analysed in Section 3.2.4. In order to make all clearer, we unify in a single table (Table 3.8) all the information about Levenberg-Marquardt

and L-BFGS methods that we presented in Table 3.3, 3.6 and 3.7 about the smaller test problem of size $11 \times 11 \times 5$.

First of all we observe that, between the two L-BFGS methods, Algo-

Table 3.8: Statistics of various methods for the test problem of size $11 \times 11 \times 5$.

L-BFGS Method, Algorithm 6				
	err	*k*		*time*
$\mu = 3.162 \cdot 10^2$	$2.134439e-02$	18		30.0
L-BFGS Method, Algorithm 8				
	err	*k*		*time*
$\mu = 3.162 \cdot 10^{-2}$	$2.160192e-02$	16		25.7
Levenberg-Marquardt Method				
	err	*k*	*itCG*	*time*
$\mu = 10^2$	$2.111874e-02$	5	16	16.0
$\mu = 0$	$2.204699e-02$	5	19	19.0
$\mu_k = \max\left\{10^{-2},\ \min\left\{10^2, \|J_k^T r_k\|\right\}\right\}$	$2.116727e-02$	5	17	17.0

rithm 6 reaches the lowest error value, but it needs more iterations and time. Analysing all the methods together, from Table 3.8 we note that, even if the Levenberg-Marquardt method with $\mu = 10^2$ reaches the lowest error and the L-BFGS method via Algorithm 8 the highest one, all the methods tested have roughly the same relative error. Moreover, since the computational cost of the Levenberg-Marquardt and L-BFGS methods is comparable, we observe that in general the L-BFGS methods carry out less iterations than Levenberg-Marquardt method but more time is necessary; on the other hand, L-BFGS algorithms have the advantage in occupying a limited quantity of storage.

Now we move to making the same analysis for the bigger test problem, namely of size $31 \times 31 \times 7$. We write the relative error, the iterations number and the time value in Table 3.9 and we report in the same table also

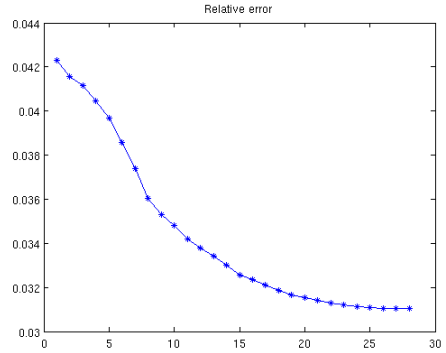
Table 3.9: Statistics of various methods for the test problem of size $31 \times 31 \times 7$.

L-BFGS Method, Algorithm 6				
	err	k		$time$
$\mu = 3.162 \cdot 10^2$	$3.104404e-02$	27		4725
L-BFGS Method, Algorithm 8				
	err	k		$time$
$\mu = 3.162 \cdot 10^{-2}$	$3.305237e-02$	100		16840
Levenberg-Marquardt Method				
	err	k	$itCG$	$time$
$\mu = 10^2$	$3.091459e-02$	8	36	1607
$\mu = 0$	$3.640818e-02$	4	22	828
$\mu_k = \max\left\{10^{-2},\ \min\left\{10^2, \|J_k^T r_k\|\right\}\right\}$	$3.214635e-02$	7	31	1346

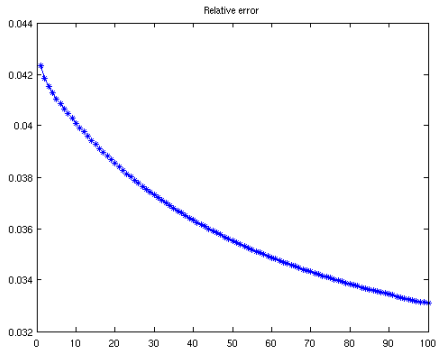
the information for the Levenberg-Marquardt method relative to the same problem written in Table 3.4.

If, as we said by watching Figure 3.16 for the test problem of size $11 \times 11 \times 5$, $\mu = 3.162 \cdot 10^{-2}$ was a good value for Algorithm 8 in order to reach a small relative error in not so many iterations, from Table 3.9 we can note that this value is not efficient for the test problem of size $31 \times 31 \times 7$, because Algorithm 8 ends reaching the maximum number of iterations. On the other hand, the value $\mu = 3.162 \cdot 10^2$ seems to be a good value for Algorithm 6 also in the bigger test problem, because it ends in 27 steps and with a quite small relative error. From Figure 3.18 we can see the plot of the relative errors for both the algorithms: if the curve for Algorithm 6 reaches the semi-convergence quite soon, on the contrary the error curve for Algorithm 8 decreases too slowly running out of the maximum number of iterations. For this reason, probably it is preferable Algorithm 6 to Algorithm 8.

Comparing from Table 3.9 Algorithm 6 to the Levenberg-Marquardt performance for the test problem of size $31 \times 31 \times 7$ we can observe that the former



(a) Plot of the relative error for Algorithm 6 and $\mu = 3.162 \cdot 10^2$ as a function of the external iterations.



(b) Plot of the relative error for Algorithm 8 and $\mu = 3.162 \cdot 10^{-2}$ as a function of the external iterations.

Figure 3.18: Relative error curves for Algorithm 6 and 8 for the problem of size $31 \times 31 \times 7$

reaches a very good relative error, roughly near to error of the Levenberg-Marquardt method for $\mu = 10^2$ constant. Moreover, even if it needs about three times as much time necessary to the Levenberg-Marquardt method, Algorithm 6 has also the advantage to require only a limited quantity of storage.

To conclude, we have a look on the reconstructed images in Figure 3.19. Surprisingly, although Algorithm 8 is quite slow and ends for the maximum number of iterations with an exit error value higher than the error reached by Algorithm 6, it seems that it produces a good solution, whose borders are less blurred and sharper. Nevertheless, the image reconstructed by Algorithm 6 needs less time and, compared with the images reconstructed by Levenberg-Marquardt methods in Figure 3.9, it seems to have roughly the same quality.

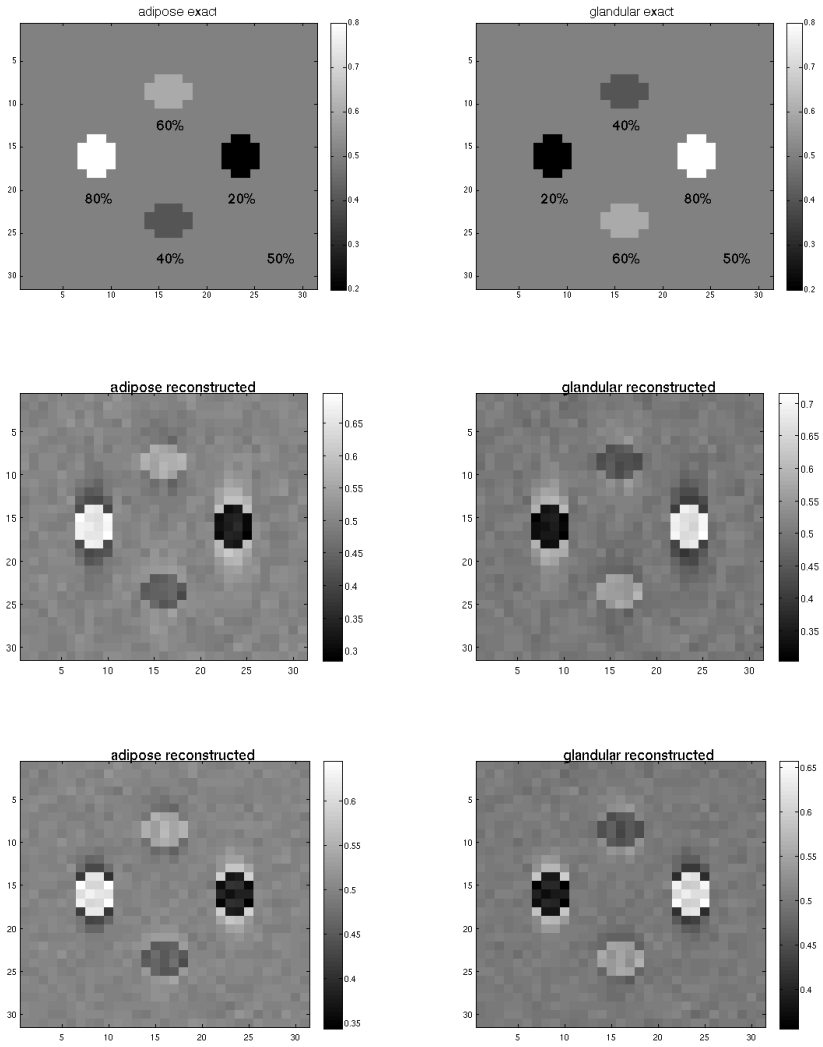


Figure 3.19: Reconstructed images for L-BFGS methods and the test problem of size $31 \times 31 \times 7$.
1st row: exact images. 2nd row: reconstructed images for Algorithm 6 with $\mu = 3.162 \cdot 10^2$. 3rd row: reconstructed images for Algorithm 8 with $\mu = 3.162 \cdot 10^{-2}$.

Conclusions

In this thesis we have analysed a polyenergetic and multimaterial model for the breast image reconstruction in Digital Tomosynthesis, namely such that the test object has been modelled by the combination of more than one material (two materials in our experiments) and such that the X-rays beam has been considered at several energy levels. The modelling of the problem has lead to the resolution of a high-dimensional nonlinear least-squares problem that, due to its nature of inverse ill-posed problem, needs some kind of regularization.

We tested two main classes of methods: the Levenberg-Marquardt method (together with the Conjugate Gradient method for the computation of the descent direction) and two methods from the limited-memory BFGS class (L-BFGS).

For the first method we made some experiments for different values of the regularization parameter (constant or varying at each iteration), tolerances and stop conditions. From the numerical results we conclude that probably the best performances correspond to constant μ value of order 10^2, varying μ as in (3.4) or with no regularization at all ($\mu = 0$) for not so small CG tolerance values (for example $tol = 5 \cdot 10^{-1}$).

From the numerical results of the two methods of the L-BFGS class we can observe that the best choice for the μ parameter is to take it constant;

moreover, we note that for the smaller problem both strategies have a good performance but, for the bigger one, Algorithm 6 seem to be much more efficient than Algorithm 8.

From an overall perspective, we can conclude that between the numerical methods we analysed the most performing ones seem to be the Levenberg-Marquardt method with a constant value of the regularization parameter for high CG tolerance values and the L-BFGS method given by Algorithm 6 again with μ constant. As regards the errors, the former seems to have a lower relative error than the latter. On the other hand, even if the L-BFGS method given by Algorithm 6 takes more time in order to reach the solution, it has the advantage of needing only a limited quantity of storage. After all, with regard to the qualities of the reconstructed images, both methods seem to be comparable.

Bibliography

[1] J.Nocedal, S. J. Wright, *Numerical Optimization*, 2nd ed. Springer (2006).

[2] J. B. Erway, R. F. Marcia, *Limited-memory BFGS systems with diagonal updates*, Linear Algebra and its Applications, 437 (2012), pp 333-344.

[3] J. G. Nagy, *A multi-material model for polyenergetic digital breast tomosynthesis reconstruction*, Technical Report (2014).

[4] V. M. Bustamante, J. G. Nagy, S. S. J. Feng and I. Sechopoulos, *Iterative breast tomosyntesis image reconstruction*, SIAM J. Scientific Computing 35(5) (2013), pp S192-S208.

[5] R. L. Siddon, *Fast calculation of the exact radiological path for a three-dimensional CT array*, Medical Physics, 12, 252 (1985).

[6] J. Chung, J. Nagy, I. Sechopoulos, *Numerical algorithms for polyenergetic digital breast tomosyntesis reconstruction*, SIAM J. Imaging Sci., Vol 3, No. 1 (2010), pp 133-152.

[7] G. Landi, E. Loli Piccolomini, J. G. Nagy, *Numerical solution of a nonlinear least squares problem in digital breast tomosynthesis*, submitted to Journal of Physics, Conference Series (2015).

[8] M. Hanke, *A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems*, Inverse Problems, 13, 79 (1997).

[9] L. Grippo, M. Sciandrone, *Metodi di ottimizzazione non vincolata*, Springer Verlag Italia (2011).