ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

A MULTILEVEL GRADIENT METHOD FOR OPTIMIZATION PROBLEMS

Tesi di Laurea in Ottimizzazione Numerica

Relatore:
Chiar.ma Prof.
MARGHERITA PORCELLI

Presentata da:
FEDERICO GRILLINI

Correlatore:
Chiar.ma Prof.
ELISA RICCIETTI

Anno Accademico 2021-2022

Contents

Introduzione

Il presente elaborato è la prosecuzione dell'attività di tirocinio di due mesi, svolto presso la École Normale Supérieure di Lione, in collaborazione col gruppo di ricerca DANTE e supervisionato dalle Professoresse Elisa Riccietti e Nelly Pustelnik.

Lo stage si colloca all'interno delle fasi iniziali di un più ampio progetto coordinato dal team DANTE. Il progetto si pone come obiettivo lo studio dello schema multilivello per metodi numerici utilizzati nell'ambito della ricostruzione di immagini.

I problemi di ottimizzazione di dimensione finita di larga scala spesso derivano dalla discretizzazione di problemi di dimensione infinita. È perciò possibile descrivere il problema di ottimizzazione su più livelli discreti. Lavorando su un livello, e quindi su una dimensione, più basso di quello del problema considerato, si possono calcolare soluzioni approssimate che saranno poi punti di partenza per il problema di ottimizzazione al livello più fine. Ovviamente, livelli più bassi implicano costi computazionali più bassi. Risolvere quindi il problema di ottimizzazione in uno spazio di dimensioni ridotte fa si che si calcoli un passo molto accurato ad un costo computazionale accettabile. I metodi multilivello, già ampiamente presenti in letteratura a partire dagli anni Novanta, sfruttano tale caratteristica dei problemi di ottimizzazione per migliorare le prestazioni dei metodi di ottimizzazione standard.

L'obiettivo di questa tesi è quello di implementare una variante multilivello del metodo del gradiente (MGM) e di testarlo su due diversi campi: la risoluzione delle Equazioni alle Derivate Parziali la ricostruzione di immagini.

L'elaborato è organizzato come segue: nel primo capitolo viene illustrata la teoria dei metodi multilivello dando particolare importanza alla costruzione dei modelli "grossolani" e alle modalità di trasferimento dei dati da un livello all'altro. Al termine del capitolo viene infine presentato l'algoritmo del metodo del gradiente multilivello. Nel secondo capitolo viene presentato un problema PDE, di cui si intende trovare la soluzione tramite MGM. Viene successivamente mostrato come discretizzare il problema e il suo spazio di definizione. Nell'ultima sezione del capitolo sono riportati i risultati sperimentali ottenuti. Come sarà più avanti discusso, essi mostrano un ottimo comportamento di MGM rispetto alla implementazione classica ad un livello. Il terzo capitolo, infine, ha come punto di partenza la descrizione del modello di degradazione dell'immagine e in base a questo si rende possibile poi formulare il problema di minimo su cui vogliamo testare il nostro metodo. La funzione da minimizzare è composta a sua volta da due funzioni: una è legata alle caratteristiche del problema (termine di fedeltà), mentre l'altra, la funzione regolarizzante, può essere scelta in base alle caratteristiche dell'immagine e al metodo di risoluzione adottato. Nel nostro caso, avendo bisogno di una mappa sufficientemente regolare, si è scelta una "edge-preserving function" liscia. Caratteristica del problema è la pressenza di iperparametri. Un parametro fa parte della funzione regolarizzante mentre l'altro bilancia i due termini che compongono la funzione. Questi parametri sono stati calcolati servendosi di un algoritmo sviluppato in [15]. Il capitolo si conclude con una esposizione dei risultati ottenuti per la ricostruzione di immagini. Vedremo come, al contrario delle PDEs, MGM sia efficace solo in determinate condizioni.

A partire da questo elaborato, lo studio dei metodi multilivello, all'interno del progetto di DANTE, proseguirà concentrandosi su varianti multilivello di proximal methods. Essi sono più vantaggiosi rispetto ai tradizionali metodi di minimizzazione di primo ordine in quanto consentono di minimizzare funzioni con condizioni di regolarità più deboli. Al contempo essi permettono l'utilizzo di modelli più adatti al problema di ottimizzazione indagato.

Introduction

Large-scale finite-dimensional optimization problems often arise from the discretization of infinite-dimensional problems. While the direct solution of such problems for a discretization level is often possible using existing packages for large-scale numerical optimization, this technique typically does make very little use of the fact that the underlying infinite-dimensional problem may be described at several discretization levels; the approach thus rapidly becomes cumbersome. The multilevel methods that we explore here, are a class of algorithms which makes explicit use of this fact in the hope of improving efficiency. The use of different levels of discretization for an infinite-dimensional problem has been studied since the early 1990s. A simple first approach is to use coarser grids in order to compute approximate solutions which can then be used as starting points for the optimization problem on a finer grid.

The objective of this thesis is to implement a multilevel variant of the gradient method (MGM) and test it on two different fields: solution of Partial Differential Equations (PDEs) and image reconstruction.

The paper is organized as follows: in the first chapter, the theory of multilevel methods is explained giving special emphasis on the construction of coarse models and how to transfer data from one model to another. Finally, the MGM algorithm is presented at the end of the chapter.

In the second chapter, the PDEs problem is presented. There, the discretization process of the problem and its domain is shown. In the last section of the chapter, the experimental results obtained are illustrated. As discussed later on, the results highlight the excellent behavior of MGM.

Finally, the third chapter devotes the first few sections to the description of the image degradation model. It comes then natural to formulate the minimization problem we want solve. The function to be minimized is composed of two functions: one is bounded to the model features (fidelity term), while the other, the regularizing function, can be chosen according to the characteristics of the image and the resolution method adopted. In our case, considering the method we want to use, a regular map is needed. For that reason, an edge-preserving function with smooth potential function is chosen. The last unknowns we have to determinate are the hyperparameters of the model. One is part of the regualrization function while the other balances the data fidelity and the amount of the regularization to the observed image. These parameters were calculated through an algorithm developed in [15]. The chapter concludes with an exposition of the performance results of MGM in image restoration. We will see later that MGM is effective only under certain conditions.

Chapter 1

Multilevel Methods

Outline: Here, an exposition of a general multilevel scheme is made following the papers [1, 2, 3]. At the end of the chapter we also show a multilevel algorithm that exploits the backtracking gradient method seen in Appendix A.

1.1 Introduction

Let us consider the unconstrained optimization problem of the form

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a differentiable function, bounded below. The iterative methods we are investigating produce a sequence $\{x_k\}_{k\in\mathbb{N}}$ of iterates converging to a first-order critical point for the problem i.e. to a point $x \in \mathbb{R}^n$ such that $\nabla f(x) = 0$.

When (1.1) results from the discretization of some infinite-dimensional problem on a relatively fine grid, the solution cost is often significant. In what follows, we investigate what can be done to reduce this cost by exploiting the knowledge of alternative simplified expressions of the objective function, when available. More specifically, we assume that we know a collection of functions $\{f_i\}_{i=0}^{l}$ such that each f_i is a twice-continuously differentiable function from $\mathbb{R}^{n_i}$ to $\mathbb{R}$ (with $n_i \geq n_{i-1}$) and we set $n_l = n$ and

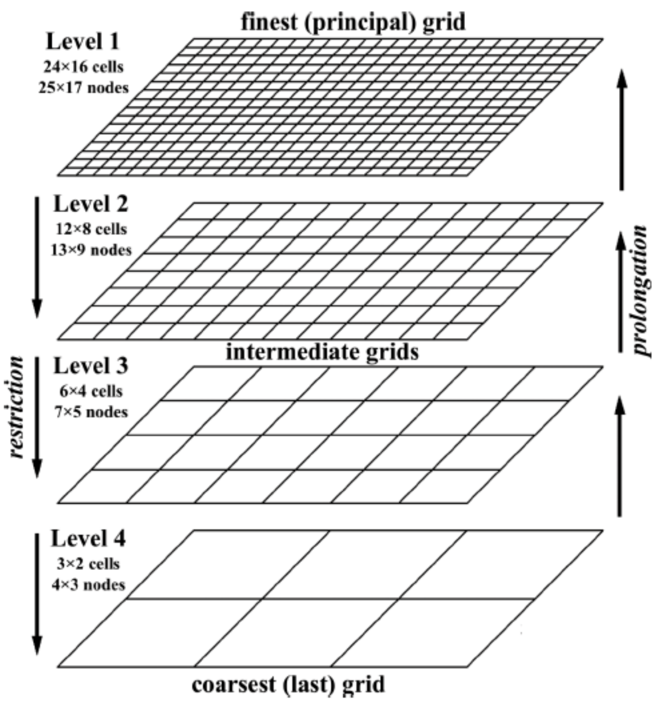


Figure 1.1: Illustration of a multilevel strategy with information transfer between fine and coarse models taken from [6].

$f_l(x) = f(x) \quad \forall x \in \mathbb{R}^n$. We will also assume that, $\forall i = 1, \dots, l$, f_i it is more costly to minimize than f_{i-1}. This may be because f_i has more variables than f_{i-1}, as e.g. if f_i represents increasingly finer discretizations of the same infinite-dimensional objective.

The main idea is then to use f_{i-1} to construct an alternative model h_{i-1} for f_i in the neighbourhood of the current iterate, that is cheaper than the model at level i, and to use this alternative model to define each step of the minimization at the i-th level. The convergence theory requires some coherence properties between f_i and its model h_{i-1}. If f_{i-1} satisfies these properties, the choice $h_{i-1} = f_{i-1}$, is possible.

If more than two levels are available ($l > 1$), this can be done recursively. When the iteration at the finer level starts, f is approximated with h_{l-1}, if it is convenient, that is used to find the step at each iterate. In order to do it, a minimum for h_{l-1} has to be found. This can be done by approximating, in turn, h_{l-1} and making a minimization on this new function.

In what follows, we use a simple notation where quantities of interest have a double subscript i, k. The first, i, $(0 \leq i \leq l)$, it is the level index and the second, k, it is the index of the current iteration within level i, and is reset to 0 each time level i is entered.

Notation 1.1. To make the notation less dense and confused, from now on we will use the following notation

$$f_{i,k} \coloneqq f_i(x_{i,k})\,.$$

1.2 Coarse model construction

Multilevel algorithms require information to be transferred between levels as in Fig. 1.1. In the proposed algorithm we need to transfer information concerning the incumbent solution, and gradient around the current point. Supposing to be at i-th level, at iteration k, the proposed algorithm projects the current solution $x_{i,k}$ from the fine level to the coarse level to obtain an initial point for the coarse model denoted by $x_{i-1,0}$. This is achieved using a suitably designed matrix R_i called *restriction operator* as follows:

$$x_{i-1,0} = R_i x_{i,k}\,. \tag{1.2}$$

In addition to the restriction operation, we also need to transfer information from the coarse model to the fine model. This is done using the *prolongation operator* P_i. The standard assumption in multigrid literature is to assume that $R_i = \sigma P_i^T$, where $\sigma \in \mathbb{R}_+$.

Assumption 1.1. Let us assume there exists two full-rank linear operators $R_i : \mathbb{R}^{n_i} \longrightarrow \mathbb{R}^{n_{i-1}}$ and $P_i : \mathbb{R}^{n_{i-1}} \longrightarrow \mathbb{R}^{n_i}$ such that $P_i = \sigma R_i^T$ for a fixed scalar $\sigma > 0$. Let us assume that it exist $\kappa_{R_i} > 0$ such that $max\{\|R_i\|, \|P_i\|\} \leq \kappa_{R_i}$, where $\|\cdot\|$ is the matrix norm induced by the Euclidean norm at the fine level.

In the following, we can assume $\sigma = 1$, without loss of generality, as the problem can easily scaled to handle the case $\sigma \neq 1$.

Now, let us see how to build the coarse model if we are at the $i-th$ level. The first task is to restrict $x_{i,k}$ to create the starting iterate $x_{i-1,0}$ at level $i-1$, that is $x_{i-1,0} = R_i x_{i,k}$. Then we define the lower model by

$$h_{i-1}(s_{i-1}) \overset{def}{=} f_{i-1}(x_{i-1,0} + s_{i-1}) + \langle v_{i-1}, s_{i-1} \rangle \,, \tag{1.3}$$

where

$$v_{i-1} = R_i \nabla h_{i,k} - \nabla f_{i-1,0} \,. \tag{1.4}$$

By convention, we set $v_l = 0$, such that, $\forall s_l \in \mathbb{R}^{n_l}$ happens that

$$h_l(s_l) = f_l(x_{l,0} + s_l) = f(x_{l,0} + s_l) \quad and \quad \nabla h_{l,k} = \nabla f_{l,k} \,.$$

The function h_i, therefore, corresponds to a modification of f_i by a linear term. It is easy to see that with the definition of v_i given above, the following first order coherency condition holds:

$$\nabla h_{i-1,0} = R_i \nabla h_{i,k} \,. \tag{1.5}$$

In fact, if s_i and s_{i-1} satisfy $s_i = P_i s_{i-1}$, then:

$$\nabla h_{i,k}^T s_i = \nabla h_{i,k}^T P_i s_{i-1} = (R_i \nabla h_{i,k})^T s_{i-1} = \nabla h_{i-1,0}^T s_{i-1} \,.$$

1.3 Step computation and step acceptance

In this section we assume to have only two levels: the finest i and the coarser $i-1$. When, at each generic iteration k in the finest level, a step $s_{i,k}$ has to be computed, to define the new iterate, the first thing to do is to choose whether is better to compute the step at the finer level or at the lower level. Obviously, it is not always possible to use the lower level model. For example, it may happen that $\nabla f_{i,k}$ lies in the nullspace of R_i, thus $R_i \nabla f_{i,k} = 0$ while $\nabla f_{i,k}$ is not. In this case, the current iterate appears to be first-order critical at lower level while it is not at higher level. Using the model h_{i-1} is hence potentially useful only if $\|R_i \nabla f_{i,k}\|$ is large enough compared to $\|f_{i,k}\|$. Therefore, we restrict the use of the lower model to iterations where the condition below is satisfied:

Condition 1.1. A coarse correction iteration is performed when both conditions below are satisfied

$$\|R_i \nabla f_{i,k}\| \geq \kappa_{i-1} \|\nabla f_{i,k}\| \tag{1.6a}$$
$$\|R_i \nabla f_{i,k}\| > \epsilon_{i-1}\,, \tag{1.6b}$$

for some constant $\kappa_{i-1} \in (0, min\{1, \|R_i\|\})$ and where $\epsilon_{i-1} \in (0,1)$ is a measure of the first-order criticality. This condition must be checked before any attempt to compute a step at a lower level.

If the fine model is chosen, then we just compute a standard iteration. While, if the lower level model is chosen, we minimize the model h_{i-1}.

Note that for the convergence of the multilevel method an approximate minimization is sufficient, so we stop when the following condition is satisfied.

$$\|\nabla h_{i-1,k}\| \leq \theta \; \|s_{i-1,k}\|\,. \tag{1.7}$$

for some $\theta > 0$.

When the iteration at the coarse level is finished, the coarse correction term $s_{i-1,k}$ is prolonged back on the fine level i.e. we define $s_{i,k} = P_l s_{i-1,k}$. Then we can update the iterate as follows: $x_{i,k+1} = x_{i,k} + s_{i,k}$.

Let see a sketch of a multilevel algorithm that is working at a generic level i. To minimize the models a simple gradient methods with the backtracking strategy is used at both the fine and coarse level.

Algorithm 1: Multilevel Gradient method (MGM)

```
Input: $x_{i,0} \in \mathbb{R}^{n_i}$,  $m \in \mathbb{N}$,  $maxit \in \mathbb{N}$,  $\kappa_{i+1} > 0$,
$\epsilon_{i+1} > 0$,  $toll > 0$,
Output: $x_{i,k}$
1: for $k = 0, 1, \dots, maxit$ do
2:     Check Assumption 1:
3:     if $\|R_i \nabla h_{i,k}\| > \kappa_{i-1} \|\nabla h_{i,k}\|$ and $\|R_i \nabla h_{i,k}\| > \epsilon_{i-1}$ then
4:         Set $x_{i-1,0} = R_i x_{i,k}$;
5:         Construction of the coarse model:
6:         $v_{i-1} = R_i \nabla h_{i,k} - \nabla f_{i-1,0}$;
7:         $h_{i-1}(s_{i-1}) = f_{i-1}(x_{i-1,0} + s_{i-1}) + \langle v_{i-1}, s_{i-1} \rangle$;
8:         Set $s_{i-1,0} = 0$;
9:         Perform the minimization of the coarse model:
10:        for $j$=0,...,$m$ do
11:            $p_{i-1,j} = \nabla h_{i-1,j}$;
12:            $s_{i-1,j+1} = s_{i-1,j} - \beta_j p_{i-1,j}$;
13:            where $[\beta_j, IND] = backtracking(h_{i-1}, s_{i-1,j}, \beta_0, p_{i-1,j})$;
14:            if IND=-1 then ◁ Check if backtracking failed
15:
16:                Prolong the step length:
17:                $s_{i,k} = P_i s_{i-1,j}$;
18:                Break;
19:            if $\|\nabla h_{i-1,j+1}\| \leq \theta \ \|s_{i-1,j+1}\|$ then
20:                $s_{i,k} = P_i s_{i-1,j+1}$;
21:                Break;
22:        $s_{i,k} = P_i s_{i-1,m}$;
23:    else
24:        $s_{i,k} = -\nabla h_{i,k}$;
25:    Update the current solution;
26:                $x_{i,k+1} = x_{i,k} - \alpha_k s_{i,k}$;
27:    where $[\alpha_k, IND] = backtracking(h_i, x_{i,k}, \alpha_0, s_{i,k})$;
28:    if IND=-1 then
29:        Stop;
30:    if $\|\nabla f_i(x_{i,k+1})\| \leq toll$ then
31:        Stop;
```

Chapter 2

Multilevel gradient method for a PDE problem

In the this chapter we explore how the multilevel gradient method, discussed in Chapter 1, behaves in finding the solution of a particular partial differential equation.

Outline: This chapter is structured as follows: we introduce our problem with its assumptions. In the next three sections we discuss how to discretize the proposed PDE (see [8]). Then, we present the problem formulation and the coarse model construction. In the last section we report the numerical results where the Multilevel Gradient Method (MGM) is compared with the standard gradient method (GM) with backtracking and some other higher order methods.

2.1 Introduction

Definition 2.1. The Laplace operator in of $u : \mathbb{R}^n \longrightarrow \mathbb{R}$, calculated in a point x_0 where u is twice differentiable, is given by

$$\Delta u(x_0) \coloneqq \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}(x_0)\,.$$

Assumption 2.1. First of all let us set what is needed to define our problem:

- $\Omega \subset \mathbb{R}^d$ is an open subset of $\mathbb{R}^d$;
- $u : \mathbb{R}^d \longrightarrow \mathbb{R}^d$, is supposed to be a twice-differentiable function;
- $g : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ is a function.

Under these assumptions we can define the following mildly nonlinear elliptic PDE:

$$\begin{cases} -\Delta u(x) + e^{u(x)} = g(x) & in \quad \Omega \\ u(x) = 0 & on \quad \partial\Omega \end{cases} \tag{2.1}$$

Assumption 2.2. We suppose to have the analytical solution of (2.1) $u^* : \mathbb{R}^d \longrightarrow \mathbb{R}^d$, so that the function $g : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ can be determined by substituting u^* in the equation.

In this section we tackle two instances of the problem:

$d = 1 \qquad u^*(x) = cos(2\pi x(x-1)) - 1, \quad \Omega = (0,1)$;

$d = 2 \qquad u^*(x_1, x_2) = sin(2\pi x_1(1-x_1))sin(2\pi x_2(1-x_2)), \quad \Omega = (0,1)\times(0,1).$

2.2 Space discretization

Foremost, in order to use the MGM to solve (2.1), we need to rewrite the problem in a discrete form. So, let us see how to do it for either cases $d = 1$ and $d = 2$. For $d = 1$ the domain of the problem $\Omega = [0, 1]$ is partitioned into n subintervals by introducing the grid points $x_j = jh$, where $j = 0, \dots, n$ and $h = \frac{1}{n}$ is the constant width of the subintervals. This establishes the grid shown in Fig. 2.1, which we denote by Ω^h.

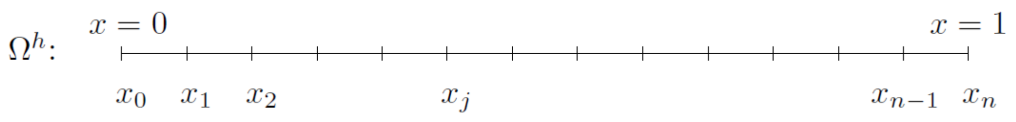


Figure 2.1: One-dimensional grid on the interval $\Omega = [0, 1]$. The grid spacing is $h = \frac{1}{n}$ and the j-th grid point is $x_j = jh$ for $j = 0, \dots, n$.

As before, for $d = 2$, the problem may be cast in a discrete form by defining the grid points $(x_i, y_i) = (ih, jh)$, where $h = \frac{1}{n}$ and $i, j = 1, \dots, n$. This two-dimensional grid is, likewise, denoted by Ω^h and is shown in Fig. 2.2.

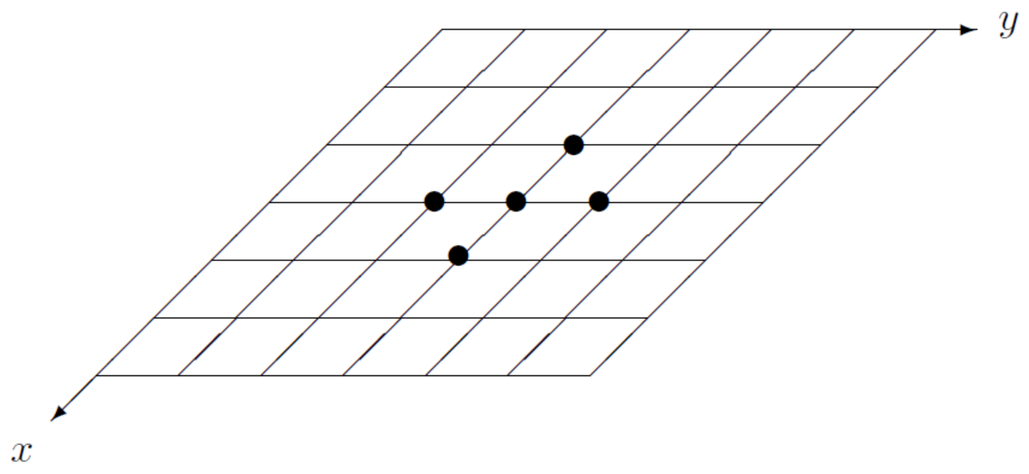


Figure 2.2: Two-dimensional grid on the unit square. The solid dots indicate the unknowns that are related at a typical grid point by the equations (2.5).

2.3 Finite differences method

The last thing left before the multilevel method can be applied is how to write the laplacian of u. In both dimensions, the negative Laplacian is written replacing the derivatives by second-order finite differences, giving a symmetric positive definite matrix $A \in \mathbb{R}^{n^d \times n^d}$, that also take into account the boundary conditions.

Theorem 2.1 (Taylor's theorem)
Let the function $u : \mathbb{R} \longrightarrow \mathbb{R}$ be twice continuously differentiable at the point $x_0 \in \mathbb{R}$. Then, there exists $R : \mathbb{R} \longrightarrow \mathbb{R}$, $\lim_{h \to 0} \quad R(x+h) = 0$, such that

$$u(x_0 + h) = u(x_0) + u'(x_0)h + \frac{u''(x_0)}{2}h^2 + R(x_0 + h)h^2 \,,$$

with $h \in \mathbb{R}$ and $|h| << 1$.

d=1 case Exploiting the Taylor's theorem and forgetting for a moment of R, when $h \to 0$ the second derivatives of f in x_0 can be written as follows:

$$\begin{aligned} u(x_0 + h) &\approx u(x_0) + u'(x_0)h + \frac{u''(x_0)}{2}h^2 \,, \\ u(x_0 - h) &\approx u(x_0) - u'(x_0)h + \frac{u''(x_0)}{2}h^2 \,. \end{aligned}$$

Multiplying the first equation by a and the second by b ($a, b \in \mathbb{R}$), and then summing up we obtain

$$au(x_0 + h) + bu(x_0 - h) \approx (a + b)u(x_0) + (a - b)u'(x_0)h + (a + b)\frac{u''(x_0)}{2}h^2.$$

We want to vanish the first derivative and the coefficient of the second derivative to be 1, hence:

$$\begin{cases} a - b = 0 \\ a + b = 2 \end{cases} \Longrightarrow \begin{cases} a = 1 \\ b = 1 \end{cases} .$$

Then, we have that

$$\Delta u(x) = u''(x) \underset{h\to 0}{\approx} \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \,.$$

This approximation is usually expressed via the following stencil

$$\Delta_h = \frac{1}{h^2} \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} . \tag{2.2}$$

d=2 case Each second partial derivative needs to be approximated similarly to the $d = 1$ case.

$$\begin{aligned} \Delta u(x,y) &= \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) \\ \underset{h\to 0}{\approx} & \frac{u(x-h,y) - 2u(x,y) + u(x+h,y)}{h^2} + \frac{u(x,y-h) - 2u(x,y) + u(x,y+h)}{h^2} \\ &= \frac{u(x-h,y) + u(x+h,y) - 4u(x,y) + u(x,y-h) + u(x,y+h)}{h^2} \,, \end{aligned}$$

which is usually given by the stencil

$$\Delta_h = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} . \tag{2.3}$$

2.4 Discretized problem

Thanks to what has just been said in the Sections 2.2 and 2.3, we can write our discretized problem by replacing the derivatives by second-order finite differences. As explained before, the set Ω is dicretized using a grid of equispaced points. Hence, we have:

d=1

$$\begin{aligned} & x_j = jh \,, \quad h = \frac{1}{n} \,, \\ & u_j = u(x_j) \,, \quad j = 1, \ldots, n \,, \\ & \Delta u_j = \Delta u(x_j) \,, \quad j = 1, \ldots, n \,. \end{aligned}$$

Then, writing the vectors, we obtain:

$$\begin{aligned} u &= (u_1, \dots, u_n)^T\,, \\ u^* &= (u_1^*, \dots, u_n^*)^T\,, \\ g &= (g_1, \dots, g_n)^T = (-\Delta u_1^* + e^{u_1^*}, \dots, -\Delta u_n^* + e^{u_n^*})^T\,. \end{aligned}$$

$$\begin{cases} \frac{-u_{j-1}+2u_j-u_{j+1}}{h^2} + e^{u_j} = g_j, \quad \forall j = 0, \dots, n \\ u_0 = u_n = 0 \end{cases} \tag{2.4}$$

d=2

$$\begin{aligned} x_{i,j} &= (x_i, y_j) = (ih_x, jh_y)\,, \quad h_x = h_y = \frac{1}{n}\,, \\ u_{i,j} &= u(x_i, y_j) = u(x_{i,j})\,, \quad i,j = 1, \dots, n\,, \\ \Delta u_{i,j} &= \Delta u(x_i, y_j)\,, \quad i,j = 1, \dots, n\,. \end{aligned}$$

The grid function to be operated on should be "flattened" to a vector stacking the columns to create vectors, as

$$\begin{aligned} u &= (u_{1,1}, \dots, u_{1,n}, \dots, u_{n,1}, \dots, u_{n,n})^T\,, \\ u^* &= (u_{1,1}^*, \dots, u_{1,n}^*, \dots, u_{n,1}^*, \dots, u_{n,n}^*)^T\,, \\ g &= (g_{1,1}, \dots, g_{1,n}, \dots, g_{n,1}, \dots, g_{n,n})^T = (-\Delta u_{1,1}^* + e^{u_{1,1}^*}, \dots, -\Delta u_{n,n}^* + e^{u_{n,n}^*})^T\,. \end{aligned}$$

Now, we can finally write our discretized version of the problem

$$\begin{cases} \frac{-u_{i-1,j}-u_{i+1,j}+4u_{i,j}-u_{i,j-1}-u_{i,j+1}}{h^2} + e^{u_{i,j}} = g_{i,j} & \forall i,j = 0, \dots, n \\ u_{0,j} = u_{i,n} = u_{i,0} = u_{n,j} = 0 & \forall i,j = 0, \dots, n \end{cases} \tag{2.5}$$

2.5 Problem formulation

Expressing the systems (2.4) and (2.5) trough matrices and vectors, we have:

$$Au + e^u = g\,, \tag{2.6}$$

where u, e^u, $g \in \mathbb{R}^{n^d+1}$ and the discretized laplacian $A \in \mathbb{R}^{n^d+1} \times \mathbb{R}^{n^d+1}$ is a symmetric positive definite matrix and its pattern is described by the stencils in (2.2) and (2.3).

d=1 The laplacian is a tridiagonal matrix with -2 on the main diagonal and 1 on the lower and upper diagonal as showed in (2.2).

$$A = -\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}.$$

To take into account the boundary conditions, a column of zeros must to be added at positions 0 and n, given that $u = 0$ in $\partial\Omega$. Otherwise, it is possible to work without considering the first and the final element of u and taking $A \in \mathbb{R}^{n^d-1} \times \mathbb{R}^{n^d-1}$ like above.

d=2 As in $d = 1$, the Dirichlet boundary conditions are used. The total grid consists of $n-1$ vertical strips of length $n-1$, leaving out the values on the boundary. Thus, since the state vector u is defined as the concatenation of these vertical strips, the matrix operator A is a tridiagonal block matrix as shown in Figure 2.3; it consists of $(n-1) \times (n-1)$ blocks of dimension $(n-1) \times (n-1)$. The identity matrices in the lower and upper diagonal has to be added because of the vectorization of Ω. To see more in detail how this matrices are built, see at [5].

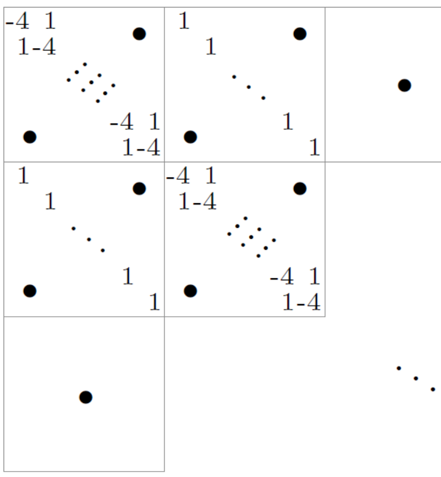


Figure 2.3: Discretized laplacian with zero boundary conditions. Values have to be scaled by $\frac{1}{h^2}$, and $\bullet$ indicates zero entries.

The following nonlinear minimization is then solved (see [3]):

$$\min_{u\in\mathbb{R}^{n^d}} \quad \frac{1}{2}u^T Au + \|e^{u/2}\|^2 - g^T u\,, \tag{2.7}$$

which is equivalent to solve the system $Au+e^u = g$. Recalling the assumption 2.2 we imposed on the PDE problem, the vector g can be computed through a more compact expression than the one in the Section 2.2: $g = Au^* + e^{u^*}$.

2.6 Coarse model

The coarse approximation of the objective function arise from a coarser discretization of the problem. Supposing to work excluding the extremities of the vectors as proposed in the previous paragraph and that the current iteration at the fine level is the k-th, to find the objective function at the coarse level it is not too hard. Calling h the finest level and H the lower, the Laplacian operator has to be built in the coarse dimension: $A_H \in \mathbb{R}^{n^d/2-1\times n^d/2-1}$. Furthermore, as in the finer level, $A_H u_H^* + e^{u_H^*} = g_H \in \mathbb{R}^{n^d/2-1}$ has to be computed. Lastly, the current iterate needs to be restricted with an operator that will be called R.

$$Rx_{h,k} = x_{H,0}, \quad x_{H,0} \in \mathbb{R}^{n^d/2-1}. \tag{2.8}$$

To restrict the current iterate and prolong the step length two linear operators have to be used. There are many possibilities in the choosing of these operators; the ones we decided to use for our problem are the linear *interpolation* operator and *full weighting* operator.

2.6.1 Interpolation operator

The linear interpolation operator will be denoted P. It takes coarse-grid vectors v^H and produces fine-grid vectors v^h, according to the rule $v^h = Pv^H$, where

d=1

$$\begin{aligned} v^h_{2j} &= v^H_j\,, \\ v^h_{2j+1} &= \frac{1}{2}(v^H_j + v^H_{j+1}), \quad 0 \le j \le \frac{n}{2} - 1\,. \end{aligned}$$

Figure 2.4 shows graphically the action of P. At even-numbered fine-grid points, the values of the vector are transferred directly from Ω^H to Ω^h. At odd-numbered fine-grid points, the value of v^h is the average of the adjacent coarse-grid values. We note also that P is a linear operator from $\mathbb{R}^{\frac{n^d}{2}-1}$ to $\mathbb{R}^{\frac{n^d}{2}-1}$. In fact, we need to restrict the iterate without the extremities, since they are zero because of the boundary condition. For the case $n = 8$, this operator has the form

$$Pv_H = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_H = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}_h = v^h\,.$$

d=2 For two-dimensional problems, the interpolation operator may be defined in a similar way. If we let $v^h = Pv^H$, then the components of v^h are given by

$$\begin{aligned}
v^h_{2i,2j} &= v^H_{i,j}\,,\\
v^h_{2i+1,2j} &= \frac{1}{2}\big(v^H_{i,j} + v^H_{i+1,j}\big)\,,\\
v^h_{2i,2j+1} &= \frac{1}{2}\big(v^H_{i,j} + v^H_{i,j+1}\big)\,,\\
v^h_{2i+1,2j+1} &= \frac{1}{4}\big(v^H_{i,j} + v^H_{i+1,j} + v^H_{i,j+1} + v^H_{i+1,j+1}\big)\,.
\end{aligned}$$

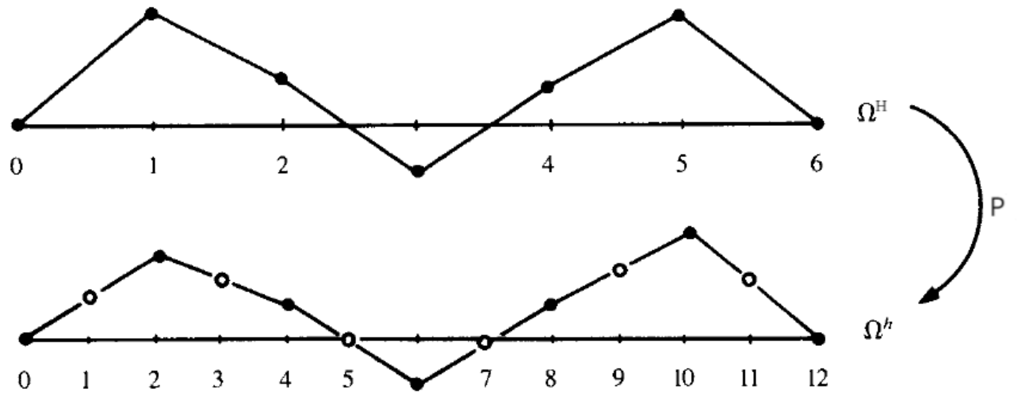


Figure 2.4: Interpolation of a vector on coarse grid Ω^H to fine grid Ω^h.

2.6.2 Restriction operator

The full weighting operator is denoted R and it is defined by $Rv^h = v^H$, where

d=1

$$v^H_j = \frac{1}{4}(v^h_{2j-1} + 2v^h_{2j} + v^h_{2j+1})\,, \quad 1 \le j \le \frac{n}{2} - 1\,.$$

As Fig. 2.5 shows, the values of the coarse-grid vector are weighted averages of values at neighboring fine-grid points. However, in some instances, *injection* may be a better choice than full weighting. The issue of intergrid transfers, which is an important part of multigrid theory, is discussed at some length in Brandt's guide to multigrid [4]. The full weighting operator is a linear operator from $\mathbb{R}^{\frac{n^d}{2}-1}$ to $\mathbb{R}^{\frac{n^d}{2}-1}$. For the case n = 8, the full weighting operator has the form

$$Rv^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}_h = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_H = v^H\,.$$

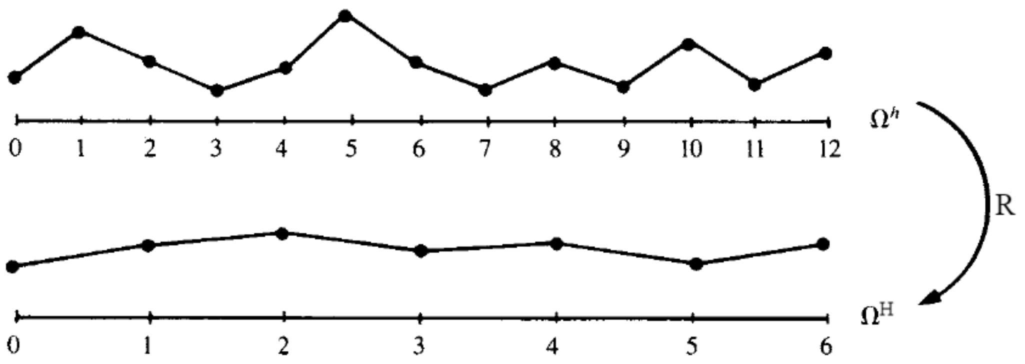


Figure 2.5: Restriction by full weighting of a fine-grid vector to the coarse grid.

d=2 For the sake of completeness, we give the full weighting operator in two dimensions. It is just an averaging of the fine-grid nearest neighbors. Letting $Rv^h = v^H$, we have that

$$\begin{aligned} v^H_{ij} =& \frac{1}{16}\Bigg(v^h_{2i-1,2j-1} + v^h_{2i+1,2j-1} + v^h_{2i-1,2j+1} + v^h_{2i+1,2j+1} + \\ &+ 2\big(v^h_{2i,2j-1} + v^h_{2i,2j+1} + v^h_{2i-1,2j} + v^h_{2i+1,2j}\big) + \\ &+ 4v^h_{2i,2j}\Bigg)\,, \quad 1 \le j \le \frac{n}{2} - 1\,. \end{aligned}$$

One of the most important reasons for our choice of full weighting as a restriction operator is the important fact

$$P = \sigma R^T\,, \qquad\qquad \sigma \in (0,+\infty)\,. \tag{2.9}$$

The fact that the interpolation operator and the full weighting operator are transposes of each other up to a constant is said *variational property.*

2.7 Numerical results

In this section we illustrate the numerical performance of the multilevel gradient algorithm (see Algorithm 1) on both $d = 1$ and $d = 2$ dimensions of the PDE problem (2.1). The optimization model used is the minimum problem exposed in Section 2.7 and reported below:

$$\min_{u\in\mathbb{R}^{n^d}} \quad \frac{1}{2}u^T A u + \|e^{u/2}\|^2 - g^T u\,.$$

To restrict the current iterate and prolong the step, the full weighting operator R and linear interpolation operator P, seen in the two Sections 2.6.1 and 2.6.2, are used. To build P we have to compute the Kronecker product

$$P = P_1 \otimes P_1\,,$$

where $P_1 = \begin{bmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix}$.

Then, as usual, thanks to the variational property (2.9) we can easily derive R:

$$R = \frac{1}{4}P.$$

For the backtracking technique we set $c_1 = 10^{-4}$, $\gamma = 0.5$, $b_{max} = 20$. The condition to use the coarse model in MGM is specified in 1.1 and we used $\kappa_i = 0,1$ and $\epsilon_i = 10^{-7} \quad \forall i = 1 \ldots l_{max}$ in our implementation. For the algorithm to terminate, stopping criteria must be set. Here is a summary:

- Maximum number of iteration "*maxit*": it varies in according to the number of levels we decided to use.

- Relative error: if x_k is the current iterate we end up the iteration when
$$\frac{\|x_k - x_{k-1}\|}{\|x_{k-1}\|} \leq \epsilon \, .$$
Analogously to coarse model condition we fix $\epsilon = 10^{-10}$.

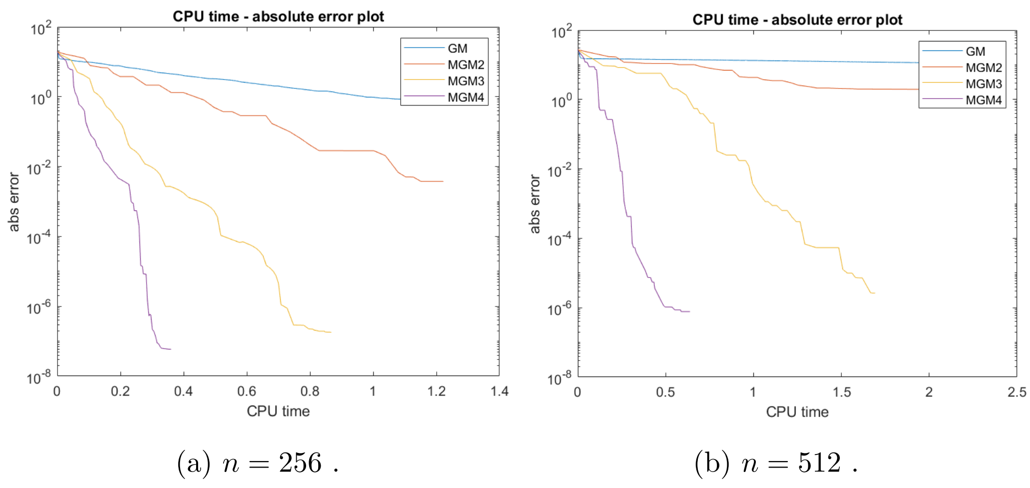
- Backtracking failure: Whether throughout its whole iteration the backtracking technique didn't succeed in finding a stepsize that satisfies both (A) and (W), then, the stepsize would be too small and therefore it would be useless to calculate a new iterate.

We saw empirically that good values for *maxit* are:

$d = 1$: When we have four level $maxit_i = 5$ with $i = 1, 2, 3$, where i indicates the level. When we have three levels instead, we set $maxit_2 = 20$ and $maxit_1 = 15$. With only two levels we have to perform many iteration in the coarse model to obtain a good step. For that reason we usually fix $maxit_1 = 200$.

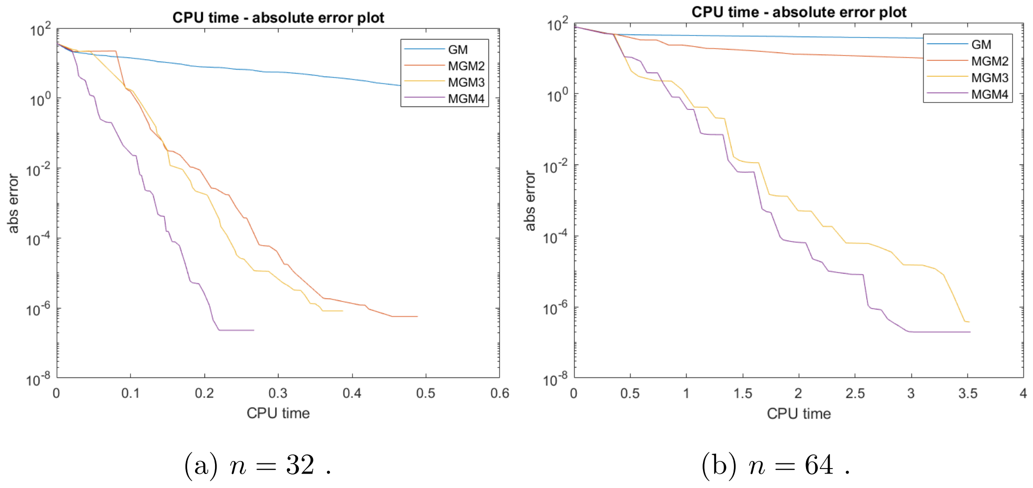
$d = 2$: Similar to the case $d = 1$, for $l_{max} = 4$ we have $maxit_i = 5, \quad i = 1, 2, 3,$ for $l_{max} = 3$, $maxit_1 = 5$, $maxit_2 = 10$ and when there are only two levels $maxit_1 = 20$.

We compare the performance of MGM with a first and a second order method. For both orders we use Algorithm 3 in Appendix A. The descent direction chosen for the first order method is $p_k = -\nabla f(x_k)$ (GM) while for the second order the descent direction p_k^N (NM) is defined by the Newton's system (A.4). We study the effect of the multilevel strategy on the convergence of the method for problems of fixed dimension n. We consider the solution of problem (2.1) in case d = 1 for $n = 256$ and $n = 512$, and in case $d = 2$ for $n = 32$ and $n = 64$.



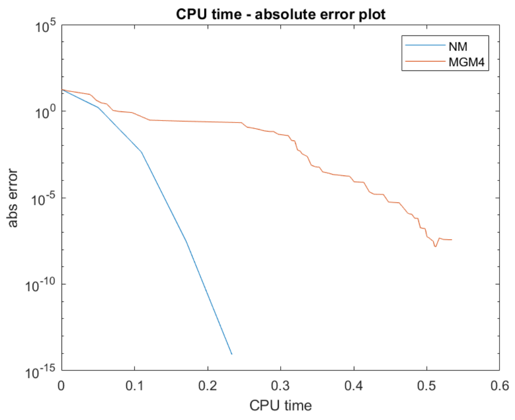
(a) $n = 256$. (b) $n = 512$.

Figure 2.6: Comparison between MGM with 2, 3 and 4 levels and GM through a CPU time - absolute error plot for $d = 1$.

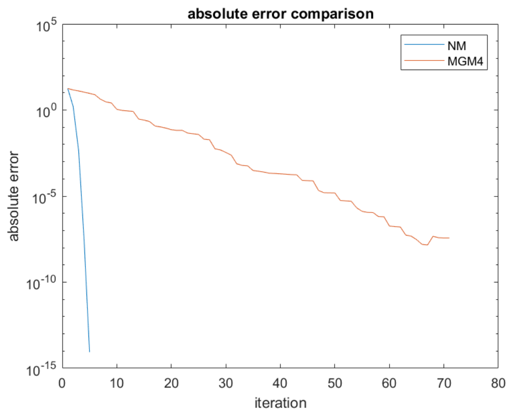


(a) $n = 32$. (b) $n = 64$.

Figure 2.7: Comparison between MGM with 2, 3 and 4 levels and GM through a CPU time - absolute error plot for $d = 2$.

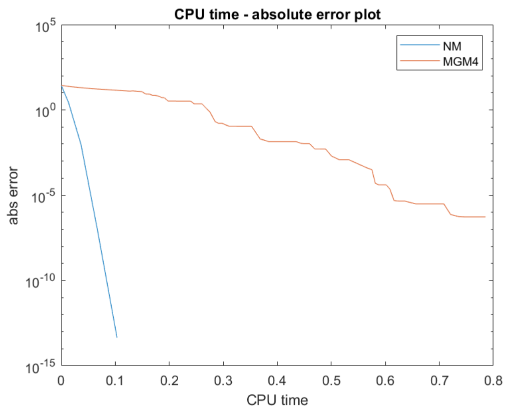


(a) iteration - absolute error

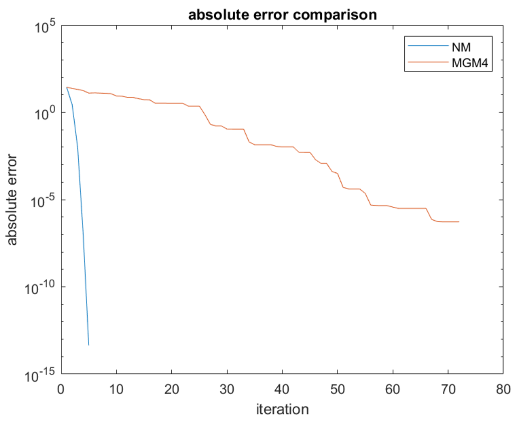


(b) CPU time - absolute error

Figure 2.8: Comparison between MGM and NM for $d = 1$ and $n = 256$.

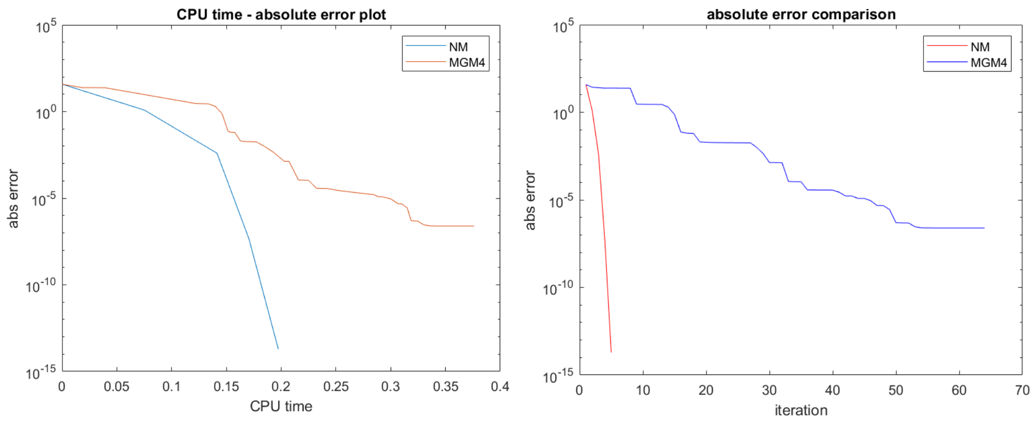


(a) CPU time - absolute error



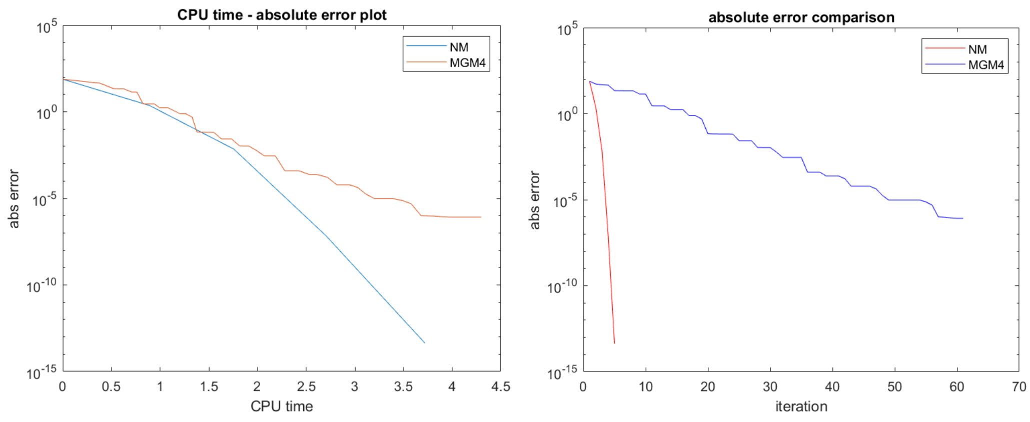
(b) iteration - absolute error

Figure 2.9: Comparison between MGM and NM for $d = 1$ and $n = 512$.



(a) CPU time - absolute error (b) iteration - absolute error

Figure 2.10: Comparison between MGM and NM for $d = 2$ and $n = 32$.



(a) CPU time - absolute error (b) iteration - absolute error

Figure 2.11: Comparison between MGM and NM for $d = 2$ and $n = 64$.

In Fig. 2.6 and Fig. 2.7 the comparison between MGM and GM is plotted in terms of the decrease of the absolute error respect to the CPU time. The numbers after MGM in the legend are referred to the number of level implemented in each multilevel algorithm. Graphs in Fig. 2.6 and 2.7 highlight how large the difference in performance is between the multilevel method and the gradient method. The plots in Fig. 2.8b, 2.9b, 2.10b and 2.11b show the decrease of the absolute error at each iteration while in 2.8a, 2.9a, 2.10a and 2.11a the absolute error is related to the CPU time. Comparing the graphs in Fig. 2.6 and Fig. 2.7, those in Fig. 2.6a with Fig. 2.6b and those in Fig. 2.7a with Fig. 2.7b no relevant differences emerge. It means that the different dimension and the different way to discretize the space in n subintervals does not affect the efficiency of MGM.

Looking at a boarder landscape than that of first-order methods we wanted to further compare our method with a second-order method, the Newton's method. From Fig. 2.8, Fig. 2.9, Fig. 2.10, and Fig. 2.11, where in MGM four levels are implemented, it is inferred that despite the excellent results given by the multilevel gradient scheme, higher order methods lead to far superior results. In facts, MGM finds a better step than GM at each iteration but it is still far from being as accurate as Newton's method. We conclude that although MGM greatly improves the performance of GM it cannot be compared with higher order methods.

Chapter 3

Image restoration problem

Image deconvolution and reconstruction belong to the class of inverse problems. They consist of recovering, from observed data, a signal/image which is the most "similar" to the original one. This constitutes a difficult task since the observed data are often degraded by various physical processes (both linear and nonlinear) during their acquisition, storage, or transmission, and they are subject to uncertainties related to the presence of random noises and the fact that the image itself is unknown.

Outline: The first section of the chapter is an overview of the image observation model problems. The following two sections lead us to the problem formulation, studying more in detail the image degradation process (blur and noise). A focus on a particular regularization function is done in Section 3.5 and an estimation algorithm is proposed to find a good value for an important parameter of the objective function in Section 3.6. Lastly, the numerical results are reported in Section 3.7.

3.1 Observation model

Generally, it is possible to describe image deconvolution and reconstruction problems by the following generic observation model (see [7]):

$$b = \mathcal{D}_\beta(Ax) \tag{3.1}$$

where

- $b \in \mathbb{R}^{n^2}$ is the vector containing the observed values, corresponding to an image of size $n \times n$,
- $x \in \mathbb{R}^{n^2}$ is the vector consisting of the unknown values of the original image of size $n \times n$ arranged in a lexicographic order,
- $A \in \mathbb{R}^{n^2 \times n^2}$ is the matrix associated to a linear degradation operator,
- $\mathcal{D}_\beta : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ models other degradations such as nonlinear ones or the effect of the noise, parametrized by β.

For many image modalities such as optical remote sensing imaging and microscopy, the observation model reduces to the linear additive noise model:

$$b = Ax + \eta \tag{3.2}$$

where A is a blurring operator corresponding to a square matrix and η is a vector of realizations of a zero-mean noise with variance β, which is often Gaussian distributed. A motion between the scene and the camera, the defocus of an optical imaging system, lense imperfections, and atmospheric turbulences lead to a blur in the acquired images. In such case, we say that an image restoration problem has to be solved.

3.2 Blurring

In order to write the observation model (3.2), it is necessary know the blurring operator A. Firstly, we introduce a continuous model of the image degradation, as proposed in [8, 9, 10].

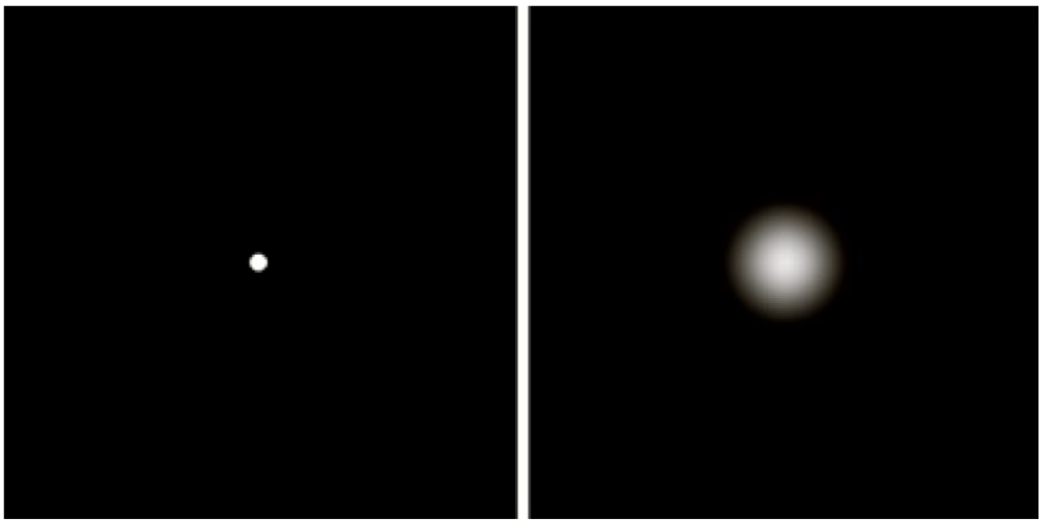


Figure 3.1: Left: single bright pixel, Right: PSF of the bright pixel.
The images are taken from [3].

Assumption 3.1. We assume that the function $X : \Omega \longrightarrow \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$, is the image which should be recorded in the absence of degradation, $g : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is the image produced by the optical instrument before detection (also called the *noiseless image*) and $b : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is the detected image. We denote by $X(y, z)$ the intensity, at the point (y, z) of the object to be imaged.

In most imaging systems the noiseless image is approximately a linear function of the object. Therefore, as in the equation (3.1) the imaging system is defined by a linear operator A such that:

$$g = AX\,. \tag{3.3}$$

If the imaging system (3.3) is isoplanatic, then it is described by a spatially-invariant *Point Spread Function* (PSF) K, i.e. the PSF is the same regardless of the location of the point source. $K(y, z)$ is the image of a point source located in the center of the image domain (see Fig. 3.1). In many examples, the light intensity of the PSF is confined to a small area around the center of the PSF (the pixel location of the point source), and outside a certain radius, the intensity is essentially zero. In other words, the blurring is a local phenomenon. Furthermore, if we assume that the imaging process captures all light and that we are working with grayscale intensity images with values in $[0,1] \subset \mathbb{R}$, then the pixel values in the PSF must sum to 1.

Assumption 3.2. $K : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is a linear and continuous operator such that

1. $K(y,z) \geq 0 \quad \forall y,z \in \mathbb{R}$,
2. $\iint_{\mathbb{R}^2} K(y,z)dy\,dz = 1$.

Under these assumptions the system (3.3) can now be rewritten:

$$g(y,z) := \iint_{(s,t)\in\Omega} K(y-s,z-t)X(s,t)\,ds\,dt\,, \tag{3.4}$$

i.e. the image g is the convolution product $K * X$. Comparing (3.3) and (3.4) we can note that A is a convolution operator.

As a consequence of the linear and local nature of the blurring, to conserve storage we can often represent the PSF using an array P of much smaller dimension than the blurred image. We refer to P as the PSF array.

In some cases the PSF can be described analytically, and thus P can be constructed from a function, rather than through experimentation. In other cases, knowledge of the physical process that causes the blur provides an explicit formulation of the PSF. When this is the case, the elements of the PSF array are given by a precise mathematical expression. For example, The PSF for blurring caused by atmospheric turbulence can be described as a twodimensional Gaussian function, and the elements of the unsealed PSF array are given by

$$p_{i,j} = exp\Bigg(-\frac{1}{2}\begin{bmatrix} i & -k \\ j & -l \end{bmatrix}^T \begin{bmatrix} s_1^2 & \rho^2 \\ \rho^2 & s_2^2 \end{bmatrix}^{-1} \begin{bmatrix} i & -k \\ j & -l \end{bmatrix} \Bigg), \tag{3.5}$$

where the parameters s_1, s_2, and ρ determine the width and the orientation of the PSF, which is centered at element (k,l) in P. Note that one should always scale P such that its elements sum to 1.

Coming back to the discrete model $g \in \mathbb{R}^{n^2}$ is still the result of the discrete convolution product of x and P. Considering x no longer as a vector but as a matrix, we have

$$g(i,j) = \sum_{i',j'=1}^{n} x(i',j')P_1(i'-i, j'-j)\,, \tag{3.6}$$

where P_1 is the matrix obtained by rotating the PSF array P, by 180 degrees. For instance, if $P \in \mathbb{R}^9$ is

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix},$$

then its rotation is

$$\begin{bmatrix} p_{33} & p_{32} & p_{31} \\ p_{23} & p_{22} & p_{21} \\ p_{31} & p_{12} & p_{11} \end{bmatrix}.$$

Equation (3.6) is given letting match P_1 with x by placing the center of P_1 over the (i,j) pixel in x. Corresponding components are multiplied and the results summed to compute g_{ij}. Continuing the previous example, we apply the procedure to compute g_{22} (x and g are 3×3 matrices). The shifting is

$$\begin{bmatrix} p_{33}\cdot x_{11} & p_{32}\cdot x_{12} & p_{31}\cdot x_{13} \\ p_{23}\cdot x_{21} & p_{22}\cdot x_{22} & p_{21}\cdot x_{23} \\ p_{31}\cdot x_{31} & p_{12}\cdot x_{32} & p_{11}\cdot x_{33} \end{bmatrix}. \tag{3.7}$$

g_{22} is the sum of all the elements of this matrix

$$g_{22} = p_{33}x_{11}+p_{32}x_{12}+p_{31}x_{13}+p_{23}x_{21}+p_{22}x_{22}+p_{21}x_{23}+p_{31}x_{31}+p_{12}x_{32}+p_{11}x_{33}\,.$$

Though, we run into an issue when we get close to the bound of the image, for example to calculate g_{21}, we have to match P_1 with x as follow

$$\begin{bmatrix} p_{33} \cdot \bullet & p_{32} \cdot x_{11} & p_{31} \cdot x_{12} & x_{13} \\ p_{23} \cdot \bullet & p_{22} \cdot x_{21} & p_{21} \cdot x_{22} & x_{23} \\ p_{31} \cdot \bullet & p_{12} \cdot x_{31} & p_{11} \cdot x_{32} & x_{33} \end{bmatrix} . \tag{3.8}$$

Hence, we have to decide how to replace the black dots above.

3.2.1 Boundary conditions

The most common technique for dealing with this missing information at the boundary is to make certain assumptions about the behavior of the sharp image outside the boundary. When these assumptions are used in the blurring model, we say that we impose *boundary conditions* on the reconstruction.

Zero boundary condition The simplest boundary condition is to assume that the exact image is black (i.e., consists of zeros) outside the boundary. This *zero boundary condition* can be pictured as embedding the image x in a larger image:

$$x_{ext} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & x & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} . \tag{3.9}$$

where the $\mathbf{0}$ submatrices represent the border of zero elements.

The zero boundary condition is a good choice when the exact image is mostly zero outside the boundary, as is the case for many astronomical images with a black background.

Unfortunately, the zero boundary condition has a bad effect on reconstructions of images that are nonzero outside the border. Hence, we must often use other boundary conditions that impose a more realistic model of the behavior of the image at the boundary but only make use of the information available, i.e., the image within the boundaries.

Periodic boundary condition The *periodic boundary condition* is frequently used in image processing. This implies that the image repeats itself (endlessly) in all directions. Again we can picture this boundary condition embedding the image x in a larger image that consists of replicas of x:

$$x_{ext} = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} . \tag{3.10}$$

Reflexive boundary condition In some applications it is reasonable to use a *reflexive boundary condition*, which implies that the scene outside the image boundaries is a mirror image of the scene inside the image boundaries.

Coming back to the 3×3 example, if we assume zero boundary conditions, then the shifting (3.8) becomes

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ p_{33} \cdot 0 & p_{32} \cdot x_{11} & p_{31} \cdot x_{12} & x_{13} & 0 \\ p_{23} \cdot 0 & p_{22} \cdot x_{21} & p_{21} \cdot x_{22} & x_{23} & 0 \\ p_{31} \cdot 0 & p_{12} \cdot x_{31} & p_{11} \cdot x_{32} & x_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} .$$

Hence,

$$g_{21} = p_{32}x_{11} + p_{31}x_{12} + x_{13} + p_{22}x_{21} + p_{21}x_{22} + p_{12}x_{31} + p_{11}x_{32} .$$

By carrying out this exercise for all the elements of g, it is straightforward to show that for zero boundary conditions, g and x are related by

$$\begin{bmatrix} g_{11} \\ g_{21} \\ g_{31} \\ g_{12} \\ g_{22} \\ g_{32} \\ g_{13} \\ g_{23} \\ g_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & & p_{21} & p_{11} & 0 & 0 & 0 & 0 \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & 0 & 0 & 0 \\ 0 & p_{32} & p_{22} & 0 & p_{31} & p_{21} & 0 & 0 & 0 \\ \hline p_{32} & p_{13} & 0 & p_{22} & p_{12} & 0 & p_{21} & p_{11} & 0 \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{21} & p_{21} & p_{11} \\ 0 & p_{33} & p_{23} & 0 & p_{32} & p_{22} & 0 & p_{31} & p_{21} \\ \hline 0 & 0 & 0 & p_{23} & p_{13} & 0 & p_{22} & p_{12} & 0 \\ 0 & 0 & 0 & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ 0 & 0 & 0 & 0 & p_{33} & p_{23} & 0 & p_{32} & p_{22} \end{array}\right] \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}, \tag{3.11}$$

which is exactly the system blurring system (3.3)

$$g = Ax\,. \tag{3.12}$$

We found that for zero boundary condition the blurring operator A is a block-tridiagonal matrix made up of PSF elements.

3.3 Noise

In addition to blurring, observed images are usually contaminated with noise. Noise can arise from several sources and can be linear, nonlinear, multiplicative, and additive. In our model (3.2), a common additive noise is considered. In this model, noise comes essentially from the following three sources:

- *Background photons*, from both natural and artificial sources, cause noise to corrupt each pixel value. This kind of noise is typically modeled by a Poisson process, with a fixed Poisson parameter, and is thus often referred to as *Poisson noise.*

- *Analog-to-digital conversion* of measured voltages result in readout noise. Readout noise is usually assumed to consist of independent and identically distributed random values; this is called *white noise.* The noise is further assumed to be drawn from a Gaussian (i.e., normal) distribution with mean 0 and a fixed standard deviation proportional to the amplitude of the noise. Such random errors are often called *Gaussian white* noise.

- The analog-to-digital conversion also results in *quantization error*, when the signal is represented by a finite (small) number of bits. Quantization error can be approximated by uniformly distributed white noise whose standard deviation is inversely proportional to the number of bits used.

In (3.2) additive noise η is an n^2 vector containing elements form a Poisson or Gaussian distribution (or a sum of both). For example, Gaussian white noise with standard deviation 0.01 is generated with the MATLAB command

```
\label{equation:noise} E = 0 . 01*randn (m, n).
```

3.4 Problem formulation

Once the image degradation process is been studied and that we are able to reproduce the system (3.2) having a prior knowledge of blur and noise, the goal is to find an estimate $\hat{x}(b) \in \mathbb{R}^{n^2}$ of the "clean" image x from the measurements b.

Let us first assume that the image formation process is noise free. The problem $b = Ax$ is said to be well-posed if it fulfills the Hadamard conditions [11] namely:

- existence of a solution, i.e. the range $ranA$ of A is equal to $\mathbb{R}^{n^2}$,
- uniqueness of the solution, i.e. the nullspace $kerA$ of A is equal to $\{0\}$,
- stability of the solution $\hat{x}$ relatively to the observation i.e.

$$\forall (b, b') \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \quad \|b - b'\| \to 0 \quad \Rightarrow \quad \|\hat{x}(b) - \hat{x}(b')\| \to 0\,.$$

If the first condition is satisfied the existence of a solution of (3.12) is guaranteed. The uniqueness condition make all the solutions to be equal since they belongs to $ker(A) = \{0\}$. The stability condition allows us to ensure that a small perturbation of the observed image leads to a slight variation of the recovered image.

In case of a full rank square matrix, a solution always exists as $ranA = \mathbb{R}^{n^2}$. Moreover, it is unique as A is injective. However, A may be ill-conditioned. Indeed, assuming A is invertible, $\hat{x}$ can be evaluated by applying the inverse degradation model to the observation b that is

$$\hat{x} = A^{-1}(Ax + b) = x + A^{-1}b\,.$$

However, if A is ill-conditioned , the inverse filtered noise $A^{-1}b$ may become very large so that its effect becomes significant. Thus, the inverse filtering amplifies the noise leading to an irregular image.

If A is not invertible and the recovery is good enough, the degraded version of the solution can be expected to be close to the observed vector b. Thus, the problem reduces to a least squares problem.

$$\hat{x} \in \underset{x\in\mathbb{R}^{n^2}}{\operatorname{argmin}} \frac{1}{2}\|Ax - b\|^2\,. \tag{3.13}$$

Studying A we find out that if $rank(A) < n^2$ the problem is under-determined, A is a "wide" matrix and the second Hadamard condition is not fulfilled. Otherwise, if $rank(A) > n^2$, A is a "tall" matrix and the first Hadamard condition is not fulfilled.

In conclusion, regardless of the rank of A, we need to stabilize the solution and guarantee its uniqueness through the following problem formulation

$$\hat{x} \in \underset{x\in\mathbb{R}^{n^2}}{\operatorname{argmin}}\left\{f_{obj}(x) := \frac{1}{2}\|Ax - b\|^2 + \lambda\phi(x)\right\}\,. \tag{3.14}$$

where

- $\|Ax - b\|^2$ ensures x to be as close as possible to b and thus is known as *data fidelity.*

- $\phi : \mathbb{R}^{n^2} \longrightarrow (-\infty, +\infty]$ is the *regularizer* or *penalty* term. The regularizer is chosen to reflect prior knowledge about the original image x. It enforces the relationship between the pixels and ensures that the recovered image is neither blurred nor noisy.

- λ is a parameter that balance the two objectives and therefore is called either *regularization parameter* or *hyper parameter.*

3.5 Edge-Preserving regularization

Knowledge of the imaging model, as we have just seen in section 3.4, is not always sufficient to determine a satisfying solution, and it is necessary to regularize the solution by imposing an a priori constraint. Mathematically, this constraint is often expressed through a *potential function.* A simple and well-known regularization supposes that images are globally smooth, and enforces a roughness penalty on the solution. A quadratic potential function yields oversmooth solutions.

A more realistic image model assumes that images are made of smooth regions, separated by sharp edges [12]. This is called *edge-preserving regularization* and requires a nonquadratic potential function.

The regularization term is defined as a sum of potentials which are, in general, functions of a derivative of the image. We consider first-order differences between pixels as showed in Fig.3.2. This leads to the following expression for the regularization term:

$$\phi(x) = \sum_{i,j=1}^{n} \varphi[(D_h f)_{ij}] + \varphi[(D_v f)_{ij}] \,, \tag{3.15}$$

where

$$(D_h x)_{ij} = (x_{i,j+1} - x_{i,j})/\delta \,,$$

and

$$(D_v x)_{ij} = (x_{i+1,j} - x_{i,j})/\delta \,. \tag{3.16}$$

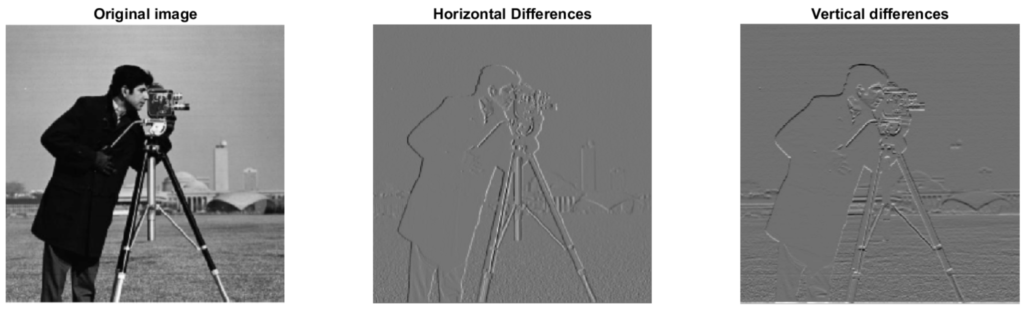


Figure 3.2: The first image is the commonly used Cameramen image of dimension 512×512 while the second and the third are respectively the horizontal and vertical differences of the original image.

D_h and D_v are the *horizontal* and *vertical differences*, respectively, while δ is a scaling parameter which tunes the value of the gradient above which a discontinuity is detected. The *potential function* φ assigns a cost to every value of the image gradient, and thus should have some obvious properties. First, in designing a potential function, it is natural to assume that φ assumes only positive values and that $\varphi(t)$ is an increasing function for $t \geq 0$. Second, it is necessary to give the same importance to gradients of equal values but opposite signs. Thus, φ is assumed to be an even function. We can then limit our study to positive values of the gradient. In order to avoid introducing instability into the reconstruction process, differentiability is desirable for φ. In addition, we have to figure out what properties should the potential function satisfy to define an edge-preserving regularization. Suppose that the objective function in (3.14) has a minimum in x, then we have necessarily

$$\frac{1}{2} f'_{obj}(x) = 0\,. \tag{3.17}$$

A simple calculation (that we skip) shows that, (3.17) can be written as

$$A^T A x - A^T b - \lambda \Delta_{pond} x = 0\,, \tag{3.18}$$

where Δ_{pond} is a matrix that represents a weighted discrete approximation of the Laplacian operator.Hence, the matrix-vector multiplication $\Delta_{pond} x$ is equivalent to a nonstationary filtering of by a 3×3 weighted Laplacian filter,

$$\begin{pmatrix} 0 & \lambda^{\mathcal{N}} & 0 \\ \lambda^{\mathcal{W}} & -\Sigma & \lambda^{\mathcal{E}} \\ 0 & \lambda^{\mathcal{S}} & 0 \end{pmatrix}$$

$\lambda^{\mathcal{E}} = \frac{\varphi'\left(x_{i,j+1}-x_{i,j}\right)}{2\left(x_{i,j+1}-x_{i,j}\right)}$	$\lambda^{\mathcal{W}} = \frac{\varphi'\left(x_{i,j}-x_{i,j-1}\right)}{2\left(x_{i,j}-x_{i,j-1}\right)}$
$\lambda^{\mathcal{S}} = \frac{\varphi'\left(x_{i+1,j}-x_{i,j}\right)}{2\left(x_{i+1,j}-x_{i,j}\right)}$	$\lambda^{\mathcal{N}} = \frac{\varphi'\left(x_{i,j}-x_{i-1,j}\right)}{2\left(x_{i,j}-x_{i-1,j}\right)}$
$\Sigma = \lambda^{\mathcal{W}} + \lambda^{\mathcal{N}} + \lambda^{\mathcal{S}} + \lambda^{\mathcal{E}}$	

Figure 3.3: Coefficients of the weighted Laplacian around pixel (i, j) (first-order neighborhood).

shown in Fig. 3.3. The weights are given by the function $\varphi'(t)/2t$, which we call the *weighting function*. Now, let us consider the case of a homogeneous area of the image: All gradients around pixel (i, j) are close to zero. Suppose that the weighting function is such that

$$\frac{\varphi'(t)}{2t} \underset{t\to 0^+}{\longrightarrow} M < +\infty\,. \tag{3.19}$$

Then, all weights around pixel (i, j) are approximately equal to M and the weighted Laplacian behaves as the usual Laplacian giving the usual normal equations associated with Tikhonov regularization (see [13])

$$A^T Ax - A^T x - \lambda M \Delta x = 0\,, \tag{3.20}$$

where Δx is discrete Laplacian of x. Now, supposing there is a discontinuity in the neighborhood of pixel (i, j), for example in between pixel (i, j) and pixel $(i, j - 1)$. Then all the finite differences around pixel (i, j) are small, except $x_{i,j-1} - x_{i,j}$. Suppose that the weighting function is such that

$$\frac{\varphi'(t)}{2t} \underset{t\to +\infty}{\longrightarrow} 0\,. \tag{3.21}$$

Then, the corresponding weighting of the Laplacian vanishes and there is no smoothing in this direction. Lastly, we suppose that $\frac{\varphi'(t)}{2t}$ is continuous, because we do not want a small variation of the gradient to produce a large change in the value of the weight. Also, it seems natural that there should be a one-to-one correspondence between values of the gradient and values of the weight. Therefore the weighting function must be strictly monotonous.

All the discussed conditions we impose on φ are summarized in the following assumption.

Assumption 3.3. Basic assumptions:

1. $\varphi(t) \geq 0 \quad \forall t$ with $\varphi(0) = 0$.

2. $\varphi(t) = \varphi(-t)$.

3. φ is continuously differentiable.

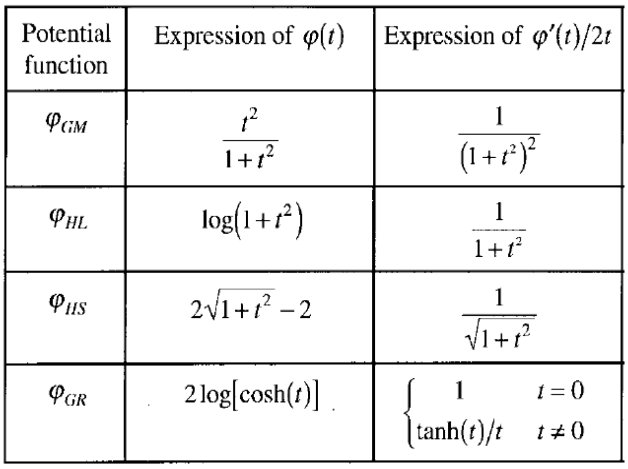
4. $\varphi' \geq 0 \quad \forall t \geq 0$.

Edge preservation:

5. $\varphi'(t)/2t$ continuous and strictly decreasing on $[0, +\infty)$.

6. $\lim\limits_{t \to +\infty} \frac{\varphi'(t)}{2t} = 0$.

7. $\lim\limits_{t \to 0^+} \frac{\varphi'(t)}{2t} = M, \quad 0 < M < +\infty$.



Potential function	Expression of $\varphi(t)$	Expression of $\varphi'(t)/2t$
φ_{GM}	$\frac{t^2}{1+t^2}$	$\frac{1}{(1+t^2)^2}$
φ_{HL}	$\log(1+t^2)$	$\frac{1}{1+t^2}$
φ_{HS}	$2\sqrt{1+t^2}-2$	$\frac{1}{\sqrt{1+t^2}}$
φ_{GR}	$2\log[\cosh(t)]$	$\begin{cases} 1 & t=0 \\ \tanh(t)/t & t \neq 0 \end{cases}$

Figure 3.4: Four edge-preserving potential functions and their associated weighting functions.

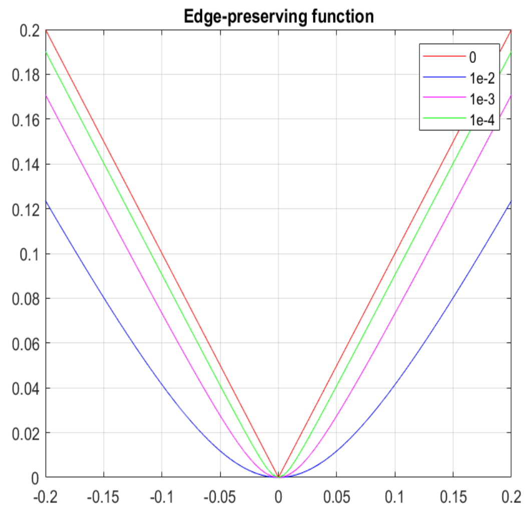


Figure 3.5: Potential function in (3.22)with different values for α .

In Fig. 3.4 we give four examples of well known potential functions and their corresponding weighting functions, but the one we use in our multilevel algorithm is

$$\varphi(t) = \sqrt{\alpha + t^2}\,, \tag{3.22}$$

where $\alpha \in \mathbb{R}_+$ is a parameter of the potential function. The function in (3.22) is a smooth approximation of the l_1 norm. The more the parameter α is small and the more the function is close to the l_1 norm as we can see in Fig. 3.5.The potential functions have been normalized in order to have $M = 1$ for all the weighting functions in Fig. 3.4.

In our discrete formulation of the imaging model, the difference operator D can be expressed as a matrix of dimension $2n^2 \times n^2$ by posing the horizontal differences matrix $D_v \in \mathbb{R}^{2n^2}$ over the vertical differences matrix $D_v \in \mathbb{R}^{2n^2}$ in this way

$$D = \begin{bmatrix} D_h \\ D_v \end{bmatrix} \qquad \text{where} \qquad D_h = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad D_v \begin{bmatrix} -1 \\ 1 \end{bmatrix}. \tag{3.23}$$

This representation of the difference operator D combined with the choice of the potential function φ as in equation (3.22) lead us to a more specific problem formulation than (3.14), that is

$$\underset{x\in\mathbb{R}^{n^2}}{\operatorname{argmin}}\left\{f_{obj}(x) := \frac{1}{2}\|Ax-b\|^2 + \lambda\phi(x)\right\}, \tag{3.24}$$

where $\phi(x) = \sum_{i=1}^{2n^2}\sum_{j=1}^{n^2}\left(\varphi[(Dx)_{i,j}]\right)$.

3.6 Regularization parameter estimation

So far we have defined the observation model (3.2) and its parts: the blurring operator in (3.11) and the noise (White Gaussian). Furthermore, we decided to solve the restoration problem by minimizing the objective function (3.24). Now we only have to set all the parameters before letting the algorithm perform.

In this section, we are going to illustrate a model, proposed in [14], to estimate the regularization parameter λ, and an algorithm is proposed to solve the model. It is critically important to have a good value of λ because when it becomes larger, the restoration image becomes smoother, and more noise is removed, but at the same time more edges are lost. On the contrary, smaller value of λ makes less noise removed and more edges preserved. So in the edge-preserving regularization method choosing a proper value of parameter λ is very important.

Our model is established based on the following conclusion. When λ is a variable, the solution $\hat{x}$ of problem (3.24) is a function of λ, which is proved in the following theorem. We denote it as $x(\lambda)$.

Lemma 3.1
When $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is strictly convex and $f \in C^1(\mathbb{R}^n)$, $\nabla f(x) = 0$ has a unique solution x^*, which is the minimizer of $f(x)$.

Lemma 3.2
Suppose that $\phi(x)$ is strictly convex. Then f_{obj}^{λ} is strictly convex.

Theorem 3.1
Suppose $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ is strictly convex, $\varphi \in C^1(\mathbb{R})$. Then $x(\lambda)$ is one-to-one correspondence, otherwise $x(\lambda) = b$ on $[0, +\infty)$.

Definition 3.1. Suppose $\min_x F(x)$, $\min_x G(x, \lambda)$ are two models for problem (3.2), where λ is a parameter of G, and $\hat{x}$, $\hat{x}(\lambda)$ are their corresponding solutions. We call $\tilde{x}$ a reference set of $\hat{x}(\lambda)$, when it is used to compute λ in

$$\min_{\lambda} \|x(\lambda) - \tilde{x}\|^2 \,. \tag{3.25}$$

For the given reference set $\tilde{x}$, there exists a solution λ^* of (3.25), where $\hat{x}(\lambda)$ is obtained by solving the problem (3.24). So the problem to estimate λ could be expressed as

$$\min_{\lambda} \|x(\lambda) - \tilde{x}\|^2 \,, \tag{3.26a}$$

$$\text{subject to} \quad 2(x(\lambda) - b) + \lambda \nabla \phi(x) = 0 \,. \tag{3.26b}$$

By the first-order necessary condition and Lemma 3.6, the constraints (3.26b) are equivalent to the problem (3.24), which ensures that $\tilde{x}(\lambda)$ is the solution of the problem (3.24). If the reference set $\tilde{x}$ is a good approximation to the original image, we can get a proper λ^* in the problem (3.24), which controls the noise removal and edge preserving. The ideal case is that the reference set $\tilde{x}$ is the original image, and the best parameter value λ can be obtained by problem (3.26). Though there are many methods to solve an equality constrained optimization problem, they do not work efficiently on our problem. Considering the specialities of problem (3.26), the authors of [12] propose a method to solve it. The one-dimensional unconstrained problem (3.25) with respect to λ is solved by Golden Section Method (Chap. 2, [15]). In this process of iterations, for each λ, $x(\lambda)$ has to be got to satisfy the constraints (3.26b). Considering the identification of (3.26b) and (3.24), we get $x(\lambda)$ by minimizing the problem (3.24), which is solved as a subiteration process by a fitting optimization algorithm (e.g. gradient method, Quasi-Newton method or Multilevel gradient method. For more details look at Chap. A and 1). The algorithm is given below.

Regularization parameter estimation algorithm

Step 1: Given the reference set $\tilde{x}$, the initial value $\lambda_0 > 0$ and $\epsilon > 0$.

Step 2: Find an interval $[\lambda_1, \lambda_2]$, which involves the solution of (3.25), by the method of Advance and Retreat (Chap. 2, [15]). That is to say, there exists $\lambda \in [\lambda_1, \lambda_2]$ which satisfies $\|\tilde{x} - \hat{x}(\lambda)\|^2 \leq \|\tilde{x} - \hat{x}(\lambda_1)\|^2$ and $\|\tilde{x} - \hat{x}(\lambda)\|^2 \leq \|\tilde{x} - \hat{x}(\lambda_2)\|^2$, where $\hat{x}(\lambda) = \underset{x}{\operatorname{argmin}} f_{obj}^{\lambda}$ and $\hat{x}(\lambda_i) = \underset{x}{\operatorname{argmin}} f_{obj}^{\lambda_i}$ for $i = 1, 2$.

Step 3: Reduce the interval $[\lambda_1, \lambda_2]$ and find the solution λ^* of problem (3.25) by the Golden Section Method (Chap. 2, [15]), until the length of the interval is less than ϵ.

Step 4: Output λ^*.

From the work done in [12] it is being found heuristically that $\|x(\lambda) - \tilde{x}\|^2$ is a unimodal function (i.e. it exists a pint such that: to the left of that point the function is monotonically increasing and to the right it is monotonically decreasing) about λ in problem (3.25), so the global minimizer λ^* of problem (3.25) can be found by the regularization parameter estimation algorithm.

3.7 Numerical results

The image restoration problem we want to to solve consists in the optimization model discussed in Section 3.4 and shown below:

$$\min_{x \in \mathbb{R}^{n^2}} \left\{ \frac{1}{2} \|Ax - b\|^2 + \lambda \phi(x) \right\},$$

where ϕ is the edge-preserving function shown in Section 3.5 and the parameter λ is found exploiting the regularization parameter estimation algorithm. The blurring operator A is based on a PSF generated by a gaussian filer of variance β and zero boundary condition (see equation (3.11)) and the image used is the cameramen image of dimensions 512×512. The interpolation

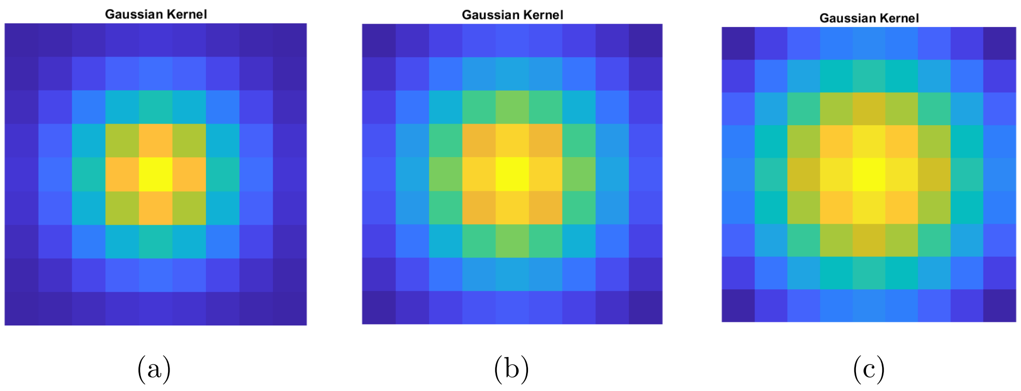


Figure 3.6: Gaussian kernels built for $\beta = 1.7,\ 3,\ 9$ from left to right respectively.

and restriction operators R and P, stopping criteria and the multilevel parameters κ and ϵ are the same as those used in PDEs in Chapter 2.

As usual, the input image and the current iterate are restricted to the coarse model by R. Instead, the blurring operator is restricted through a standard choice in multigrid theory as explained in [16]. If we call A_h the blurring operator that has to be restricted and A_H the operator at the coarse level, we have

$$A_H = R A_h P \,.$$

We set the maximum number of iteration at 20 for all the coarse levels.

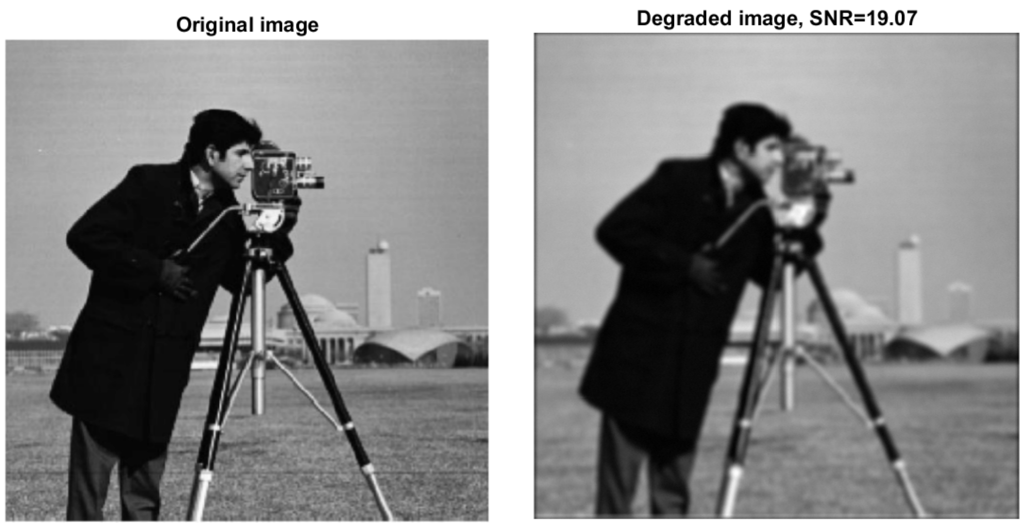
We did three tests using the same original image blurred by three gaussian kernels computed using the following values for the variance: $\beta = 1.7,\ 3,\ 9$. In addition, for $\beta = 9$ we did a further test varying the hyperparameter λ previously computed by the estimation algorithm.

We adapted the estimation algorithm to find an optimum value for the couple (λ, α), where α is the parameter of the potential function. We computed α and λ for $\beta = 9$ and found $\lambda = 0.0065$ and $\alpha = 10^{-4}$. We then set these values of λ and α for all tests. MGM2 and MGM3 in the legend represent the MGM implemented on two and three levels respectively. All the graphs are plotted in terms of the the decrease of the objective function at each iteration.

In these experiments, the noise level is evaluated with a signal-to-noise-ratio (SNR) [7] measure defined as

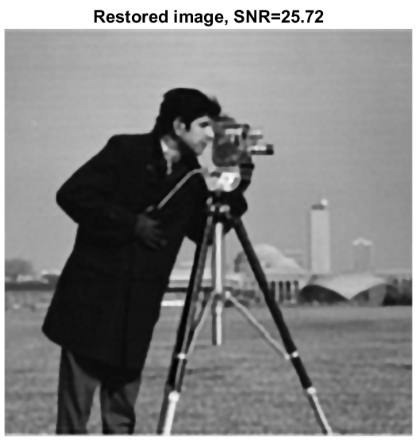
$$10\log_{10}\left(\frac{\text{variance of } x}{\text{variance of noise}}\right).$$

The higher the SNR is and the more the restored image is considered a good approximation of the original one.

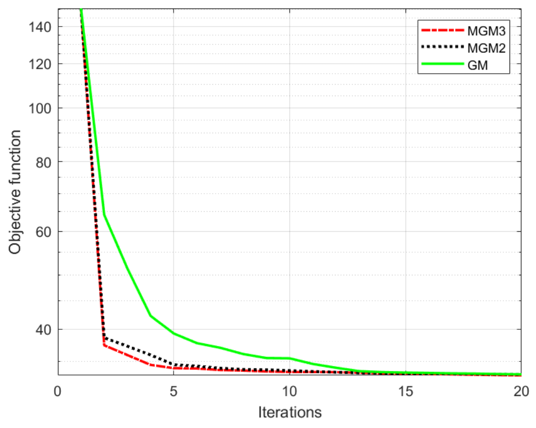


(a) Original image (b) Degraded image

Figure 3.7: Illustration of the clean image on the left and the image corrupted by a gaussian kernel for for $\beta = 9$ on the right.

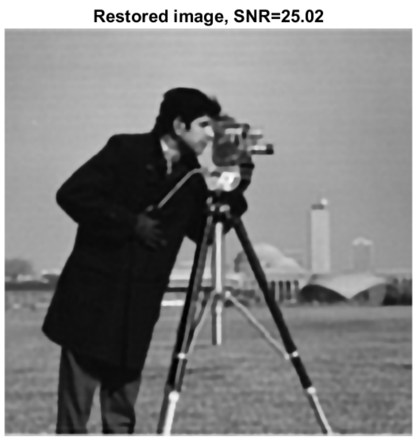


(a) Restored image

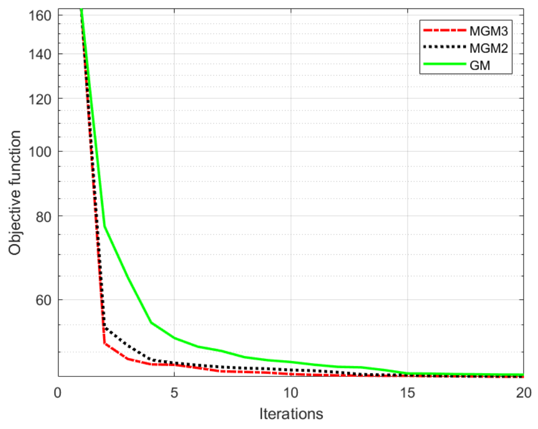


(b) Objective function

Figure 3.8: Results comparisons between MGM2, MGM3 and GM for $\beta = 9$ and $\lambda = 0.0065$.



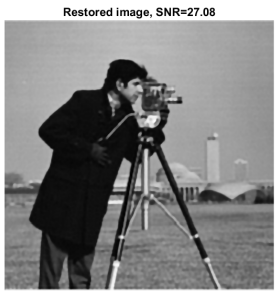
(a) Restored image



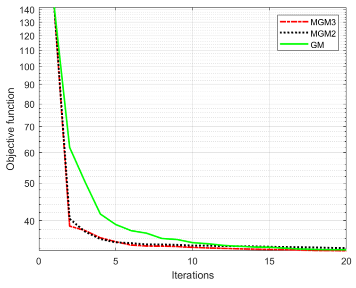
(b) Objective function

Figure 3.9: Results comparisons between MGM2, MGM3 and GM for $\beta = 9$ and $\lambda = 0.01$.

(a) Degraded image

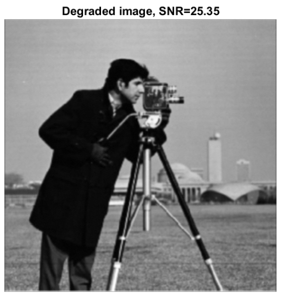


(b) Restored image

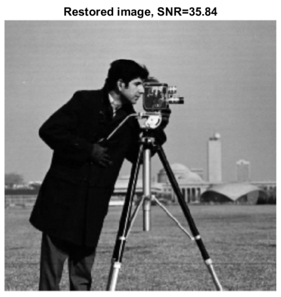


(c) Objective function

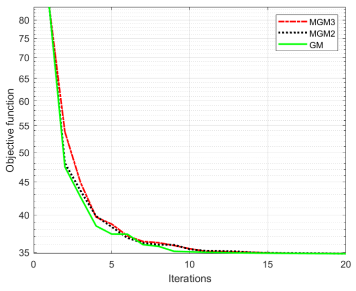
Figure 3.10: Results comparison between MGM2, MGM3 and GM for $\beta = 5$.



(a) Degraded image



(b) Restored image



(c) Objective function

Figure 3.11: Results comparison between MGM2, MGM3 and GM for $\beta = 3$.

From Fig. 3.8b, 3.9b and 3.10c it is evident that the multilvel method reduces the objective function more rapidly than GM and that MGM3 works slightly better than MGM2. However, things change when we analyze the plot in Fig. 3.11c. When the blur variance is low, GM reduces extremely well the function and the multilvel scheme is almost useless. The only drawback is that the computational cost increases and consequently the execution time increases.

An experiment has also been done on the choice of λ. MGM works efficiently both for $\lambda = 0.01$ and $\lambda = 0.0065$ as we can see in Fig. 3.8b and 3.9b but comparing the SNR values shown in Fig. 3.8a and 3.9a, it turns out that the value of λ computed by the hyperparameter estimation algorithm leads to a better restoration of the image.

Finally, it should be said that GM and MGM has been compared only in terms of decrease of the objective function and not in terms of CPU time. Since we worked with a small image of dimensions 512×512, there would have been no point in comparing the computational times. Instead, it is a reasonable measure if we consider a problem of larger dimension and perhaps try to use more levels.

Conclusions

We developed a Multilevel Gradient Method (MGM) for optimization models for solving PDEs and image restoration problems. The key idea behind MGM is to replace the fine approximation of the object function with a coarse approximation. The coarse model is used to compute search directions that are often superior to the search directions obtained using just gradient information.

For the PDE case, our numerical experiments on a mildly nonlinar elliptic PDE show that the proposed MGM algorithm is faster than the standard Gradient Method (GM) and that increasing the number of levels, improves both the speeds performance and the solution accuracy. So, it would be interesting to test it on further PDE problems.

Regarding image restoration, on the other hand, we have seen that the effectiveness of the MGM algorithm depends on the blurring operator. However, it should be pointed out that our experiments are not comprehensive. The algorithm should be tested on a large set of images and on larger dimensional problems (as mentioned in Section 3.7). Furthermore, MGM could be improved in a number of ways. For example, we only considered the most basic prolongation and restriction operators in approximating the coarse model. The literature on the construction of these operators is quite large, and there exist more accurate operators that return a better approximation of the current solution at the coarse level than the full weighting and interpolation operator used here (e.g. Wavelets can be used to prolong and restrict [8, 17, 18]). Other boundary conditions, such as the reflexive and periodic boundary conditions exposed in Section 3.2, must be taken into account as well. Additionally, the choice of the regularization term is an other important topic. For sake of simplicity, a smooth approximation of the l_1 norm (the edge-preserving function) has been used in our algorithm. However, considering new types of non-smooth penalty term, such as TV [20], make us to implement multilevel schemes on methods that can deal with non-smooth functions (e.g. Multilevel Proximal Gradient Method [19]).

Since these initial results are promising and there is a paramount panorama of possible improvements for MGM, we are hopeful that the multilevel framework can improve the numerical performance of many other algorithms in large scale optimization.

Appendix A

Line search methods for unconstrained optimization

Outline: This Appendix is a brief introduction to unconstrained optimization and was written following [21]. After an initial section where the minimum problem is posed, we focus on some basic line search directions in the next three sections. The last two sections are devoted to the backtracking strategy used to find the step length and the conditions that ensure the success of this strategy.

A.1 Introduction

Numerical methods for unconstrained optimization aim at finding the minimun of a given function f. Supposing f to be a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$, we want to solve the problem

$$\min_{x \in \mathbb{R}^n} f(x)\,. \tag{A.1}$$

Let A be a set in $\mathbb{R}^n$ and $x_0 \in A$ be a initial iterate. Iterative methods for (A.1) generally build a sequence $\{x_k\}_{k \in \mathbb{N}}$ converging to a stationary point

$x^* \in A$ of the function f, that is

$$\lim_{k \to +\infty} x_k = x^*$$

and

$$\lim_{k \to +\infty} \|\nabla f(x_k)\| = 0\,.$$

There are several ways to define x_{k+1} from a given x_k. Among these, we mention the popular *trust-region strategy* and the *line-search strategy.* This chapter is focused in the latter strategy.

Line search methods generate the iterates as follows:

$$x_{k+1} = x_k + \alpha_k p_k\,, \tag{A.2}$$

where each step $\alpha_k p_k$ is composed by two parts: the stepsize α_k and the search direction p_k. This kind of methods usually work by firstly defining the step direction, and then its length. The terms α_k and p_k are found so that

$$f(x_{k+1}) = f(x_k + \alpha_k p_k) \leq f(x_k)\,. \tag{A.3}$$

The direction can be determined in many different ways. Among these, the *steepest descent direction* is the most common one, and we will deal with it in the following section.

A.2 Gradient direction

The steepest descent direction for f in x_k is given by

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

Its name is due to the fact that it is the direction in which, starting from x_k, the value of f decrease the fastest. In fact, the steepest descent direction is the one that minimizes the directional derivative of f in x_k:

$$\frac{\partial f}{\partial p}(x_k) = \|\nabla f(x_k)\|\ \|p\|\ cos(\theta)\,,$$

where $\theta \in [0, \pi]$ is the angle between $\nabla f(x_k)$ and the vector $p \in \mathbb{R}^n$. We can assume without loss of generality that $\|p\| = 1$. Therefore minimizing the second member of the equation corresponds to take $\theta = \pi$ that gives:

$$p = \frac{-\nabla f(x_k)}{\|\nabla f(x_k)\|}\,.$$

We note that, an iterative method that use the gradient direction has a low computational cost stepwise, since it requires only the computation of the gradient of f, but at the same time the convergence is usually very slow [21].

A.3 Newton's direction

If $f \in C^2(\mathbb{R}^n)$, we can consider the quadratic model of f in x_k:

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H(x_k) p\,,$$

where the hessian of f that we call $H(x_k)$, is a positive definite matrix ($H(x_k) \succ 0$). The Newton's direction p_k^N is the global minimizer of $m_k(p)$. In fact, since $m_k(p)$ is a strictly convex quadratic function, p_k^N is the solution of the Newton's system:

$$H(x_k) p_k^N = -\nabla f(x_k)\,. \tag{A.4}$$

Thanks to the fact that $H(x_k) \succ 0$, we have that Newton's system has only one solution and also holds that p_k^N is a descent direction as shown in [20]. In fact:

$$\nabla f(x_k)^T p_k^N = \nabla f(x_k)^T \left(-H(x_k)^{-1} \nabla f(x_k)\right) = -\nabla f(x_k)^T H(x_k)^{-1} \nabla f(x_k) < 0.$$

If it was not so, the Newton's system could have more than one solution and even if p_k^N is well-defined, it may not be a descent direction.

The Newton's method is defined by setting $x_{k+1} = x_k + p_k^N$. It usually has fast local convergence, but it is expensive due to the computation of $H(x_k)$ and the solution of Newton's system at each iteration.

A.4 Quasi-Newton direction

Let us consider now a quadratic model for f of the form

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

where B_k is an approximation of $H(x_k)$ such that $B_k \succ 0$. The Quasi-Newton's direction p_k^{QN} is the minimizer of $m_k(p)$. Being $m_k(p)$ defined as a positive definite quadratic function, p_k^{QN} is the unique solution of quasi-Newton's system:

$$B_k p_k^{QN} = -\nabla f(x_k)\,. \tag{A.5}$$

A.5 Armijo and Wolfe conditions

In line search strategy after founding a direction we have to compute the step length. We would choose an α_k to have good reduction of f but at the same time we do not want to spend too much time making this choice. Furthermore, since the decrease condition (A.3) of f does not ensure the convergence of these methods, it is vital to impose a condition on the step length which guarantees the convergence.

We now introduce the conditions on the step length α_k to ensure the convergence of the overall line search method.

Armijo rule
The Armijo rule is given by the following inequality:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k c_1 \nabla f(x_k)^T p_k,\ c_1 \in (0,1)\,, \tag{A}$$

where p_k is a descent direction for f and usually $c_1 = 10^{-4}$.
Condition (A) is stronger than asking the simple decrease $f(x_{k+1}) \leq f(x_k)$ because $\nabla f(x_k)^T p_k < 0$. To better understand (A) let us call

$$
\begin{aligned}
&\phi(\alpha) = f(x_k + \alpha p_k)\,,\\
&l(\alpha) = f(x_k) + \alpha c_1 \nabla f(x_k)^T p_k\,,\\
&\phi^{'}(\alpha) = \nabla f(x_k + \alpha p_k)^T p_k\,, \quad \phi^{'}(0) = \nabla f(x_k)^T p_k < 0\,,\\
&l^{'}(\alpha) = c_1 \nabla f(x_k)^T p_k\,, \quad l^{'}(0) = c_1 \nabla f(x_k)^T p_k = c_1 \phi^{'}(0) < 0\,.
\end{aligned}
$$

Hence (A) requires $\phi(\alpha)$ to be below the line $l(\alpha)$, i.e. that $\phi(\alpha) \leq l(\alpha)$, and the slope of ϕ and l to be negatives in zero. Fixing $c_1 \in (0,1)$, it follows

$$c_1 \nabla f(x_k)^T p_k > \nabla f(x_k)^T.$$

It means that for small values of α, $l(\alpha)$ lies above the graph of ϕ.

Choosing α_k according to (A) avoids to take too large values of α_k. But it is not enough to guarantee that the algorithm makes real progress since too small step can be taken. To avoid it we need to introduce the Wolfe rule.

Wolfe rule

Wolfe rule requires that

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k, \quad c_2 \in (c_1, 1)\,. \tag{W}$$

where the first term of the inequality is the slope of $\phi(\alpha)$. The condition force $\phi^{'}(\alpha) \geq c_2 \nabla f(x_k)^T p_k$ and consequently avoids α_k to be too small. Moreover, if both (A) and (W) are required and recurring that $c_1 < c_2 < 1$ it holds:

$$c_1 \nabla f(x_k)^T p_k > c_2 \nabla f(x_k)^T p_k > \nabla f(x_k)^T p_k\,,$$

i.e. the desired slope is between those of $l(\alpha)$ and $\phi(\alpha)$.

Rules (A) and (W) allow to study the convergence of the iterative methods that use the directions shown in Sections (A.2), (A.3) and (A.4), through the following statements [21].

Lemma A.1 (Wolfe lemma)
Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$*,* $f \in C^1(\mathbb{R}^n)$ *and bounded below in* $\{x_k + \alpha p_k \ s.t. \ \alpha > 0\}$*, with* p_k *descent direction for* f *in* x_k*, and let* $c_1, c_2 : 0 < c_1 < c_2 < 1$*. Then, it exists* $I \neq \emptyset$*,* $I \subset (0, +\infty)$ *such that* $\forall \alpha \in I$ *satisfies* (A) *+* (W)*.*

Wolfe lemma guarantees the existence of at least an α satisfying both (A) and (W) conditions for a $f \in C^1(\mathbb{R}^n)$.

Theorem A.1 (Zoutendijk's theorem)
Let $\Omega = \{x \in \mathbb{R}^n \ s.t. \ f(x) \leq f(x_0)\}$*,* $f \in C^1(\Omega)$ *and lower bounded on* Ω*,* p_k *a descent direction for* f*, and assume that* α_k *satisfies* (A) *and* (W) *and that* $\nabla f(x_k)$ *is Lipschitz continuos in* Ω*. Let* θ_k *be the angle between* $-\nabla f(x_k)$ *and the vector* p_k*. Then:*

$$\sum_{j=0}^{+\infty} \left(cos(\theta_j) \|\nabla f(x_j)\|^2 \right) < +\infty \, .$$

The convergence of the series implies that

$$\lim_{k \to +\infty} cos(\theta_k) \|\nabla f(x_k)\|^2 = 0 \, .$$

It is due to two reasons, both or only one of them:

i. $\underset{k \to +\infty}{lim} \nabla f(x_k) = 0 \, .$
In this case every accumulation point of $\{x_k\}_{k \in \mathbb{N}}$, if it exists, is a stationary point;

ii. $\underset{k \to +\infty}{lim} cos(\theta_k) = 0 \, .$
It implies that

$$\lim_{k \to +\infty} \nabla f(x_k)^T p_k = 0 \, .$$

It means that $\nabla f(x_k)$ tends to be orthogonal to p_k. Therefore, along p_k f is almost constant, for $k \gg 1$. This can be avoided choosing a descent direction p_k such that $cos(\theta_k) > M$ for all k and for some $M > 0$.

Convergence of steepest descent method

Since $p_k = -\nabla f(x_k)$ we have

$$cos(\theta_k) = -\frac{\nabla f(x_k)^T p_k}{\|\nabla f(x_k)\| \|p_k\|} = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\|\nabla f(x_k)\|^2} = 1 \, .$$

Then, under the assumptions of Zoutendijk's theorem and considered that the possibility (ii) is excluded it holds: $\underset{k \to +\infty}{lim} \nabla f(x_k) = 0$.

Convergence of Newton and Quasi-Newton's method

Taken $B_k \approx H(x_k)$ the descent direction is $p_k = -B_k^{-1} \nabla f(x_k)$.

$$cos(\theta_k) = -\frac{\nabla f(x_k)^T p_k}{\|\nabla f(x_k)^T\| \|p_k\|} = \frac{\nabla f(x_k)^T B_k^{-1} \nabla f(x_k)}{\|\nabla f(x_k)\| \|B_k^{-1} \nabla f(x_k)\|} \geq \frac{\nabla f(x_k)^T B_k^{-1} \nabla f(x_k)}{\|\nabla f(x_k)\|^2 \|B_k^{-1}\|} \, .$$

To bound this quantity we need some definition and a proposition.

Definition A.1. Given $A \in \mathbb{R}^{n \times n}$, $A = A^T$, $v \in \mathbb{R}^n$ we call the Rayleigh quotient associated to the matrix A and the vector v the following:

$$r_A(v) \coloneqq \frac{v^T A v}{v^T v} \, .$$

Remark A.1. An important property of Rayleigh quotient is that $\forall v \in \mathbb{R}^n$

$$\lambda_{min}(B_k) \leq r_A(v) \leq \lambda_{max}(B_k) \, .$$

Proposition A.1

Let $A \in \mathbb{R}^{n \times n}$ or $A \in \mathbb{C}^{n \times n}$ be a symmetric positive definite matrix, and let λ_{min} and λ_{max} be respectively the minimum and maximum eigenvalue of A then:

$\|A\| = \lambda_{max}$ *and* $\|A^{-1}\| = \lambda_{max}(B_k^{-1}) = \frac{1}{\lambda_{min}(B_k)}$ *and also* $k(A) = \frac{\lambda_{max}}{\lambda_{min}}$.

Now we can conclude that:

$$\begin{aligned} cos(\theta_k) &\geq r_{B_k^{-1}}(\nabla f(x_k)) \frac{1}{\|B_k^{-1}\|} = r_{B_k^{-1}}(\nabla f(x_k)) \lambda_{min}(B_k) \\ &\geq \lambda_{min}(B_k^{-1}) \lambda_{min}(B_k) = \frac{\lambda_{min}(B_k)}{\lambda_{max}(B_k)} = \frac{1}{k(B_k)} \\ &\Rightarrow cos(\theta_k) \geq \frac{1}{k(B_k)} \, . \end{aligned}$$

Then, if it exists M such that $k(B_k) < M$, we have $cos(x_k) > \frac{1}{M}$ and under the assumptions of Zoutendijk's Theorem, it holds, $\lim_{k \to +\infty} \nabla f(x_k) = 0$ being the situation (ii) excluded again. This condition means that the approximations of the Hessian matrix have a uniformly bounded condition number (the bound is the same for any k).

A.6 Backtracking strategy

For gradient, Newton and quasi-Newton directions we know that if α_k satisfies (A)+(W) then the line-search method converges, form Theorem A.1. Therefore, remains to show how to find the right step length α_k. Several techniques can be use to do it. One of these is the backtracking strategy of which we see the algorithm below:

Algorithm 2: backtracking strategy

```
input  : $f, x_k, \alpha_0, p_k, b_{max}$, $c_1 \in (0,1), \gamma \in (0,1)$
output: $\alpha_k$, IND
for $b = 0, 1, \dots, b_{max}$ do
    if $f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \nabla f(x_k)^T p_k$ i.e. $\alpha_k$ satisfies (A) then
        IND=1;
        stop;
    else
        $\alpha_k = \gamma \alpha_k$;
    IND=-1;
```

This can be used at each iteration of the line-search algorithm. The backtracking algorithm works as follows: if $\alpha_k = \alpha_0$ does not satisfy (A), we reduce α_k by multiplying it with γ until the new α_k satisfies (A). After a finite number of reductions we will obtain an α_k for which (A) holds. So, the backtracking technique never fails. However, in the algorithm we decide to do at most b_{max} backtracking steps. If after b_{max} iterations α_k has not been found, it means that the step in the line-search is too small, which will lead to a really slow convergence.

The parameter IND is used to see if the algorithm has found an α_k satisfying (A)+(W), assigning it the value 1. Otherwise, it returns IND=-1.

The backtracking wants a starting value α_0 in input. One of the most common choice when it is used in the gradient method is done by assuming that the first-order change in the function at iteration k will be the same as that obtained at the previous iteration.

$$\alpha_0 \nabla f(x_k)^T p_k = \alpha_{k-1} \nabla f(x_{k-1})^T p_{k-1}\,,$$

$$\alpha_0 = \alpha_{k-1} \frac{\nabla f(x_{k-1})^T p_{k-1}}{\nabla f(x_k)^T p_k}\,.$$

An other popular choice is called Barzilai-Borwein

$$s_{k-1} = x_k - x_{k-1}, \quad y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})\,,$$

$$\alpha_0 = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}\,.$$

For Newton's or quasi-Newton method we choose $\alpha_0 = 1$.

We deduce then the algorithm for a line-search with backtracking technique:

Algorithm 3: Line-search with backtracking strategy

```
input  : $f, x_0, maxit, \epsilon, \alpha_0, b_{max}, c_1 \in (0,1), \gamma \in (0,1)$
output: $x_{k+1}$
for $k = 0, \dots, maxit$ do
    1: Choose the descent direction $p_k$ for $f$ in $x_k$

    2: Use the backtracking algorithm to find $\alpha_k$
       $[\alpha_k, IND] = backtracking(f, x_k, \alpha_0, p_k, b_{max}, c_1, \gamma)$;

    3: Check if the backtracking failed:
       if IND=-1 then
           return $x_k$;
    4: Update the current iterate

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

    5: Stop criterion:
       if $\|\nabla f(x_{k+1})\| \leq \epsilon$ then
           return $x_k + 1$ ;
```

Bibliography

[1] S. Gratton, A. Sartenaer and P. Toint. Recursive Trust-Region Methods for Multiscale Nonlinear Optimization. SIAM Journal on Optimization, vol. 19, iss.1, pp. 414-444, 2008

[2] Z. Wen and D. Goldfarb. A Line Search Multigrid Method for Large-Scale Nonlinear Optimization. SIAM Journal on Optimization. vol. 20, iss. 3, pp. 1478-1503, 2010.

[3] H. Calandra, S. Gratton, E. Riccietti, and X. Vasseur. On high-order multilevel optimization strategies. SIAM Journal on Optimization, pp. 307-330, 2021.

[4] A. Brandt. Multigrid Techniques: 1984 Guide with Applications to Fluid. Dynamics, GMD-Studien Nr. 85, Gesellschaft four Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1984.

[5] Bilbao, Stefan. Grid Functions and Finite Difference Operators in 2D, ch. 10, 2009.

[6] T. Gerya. Introduction to Numerical Geodynamic Modelling, Cambridge. Cambridge University Press, 2010.

[7] N. Pustelnik, A. Benazza-Benhayia, Y. Zheng, and J.-C. Pesquet. Wavelet-based image deconvolution and reconstruction. Wiley Encyclopedia of Electrical and Electronics Engineering, 2016.

[8] M. Bertero, P. Boccacci. Image Deconvolution.

[9] Per Christiam Hansen, James G. Nagy, Dianne P.O'Leary. Deblurring Images. Matrices, spectra and filtering, SIAM.

[10] M. Bertero, P. Boccacci, and C. De Mol. Introduction to Inverse Problems in Imaging. CRC Press, 2021.

[11] J. Hadamard. "Sur les problemes aux derivees partielles et leur signification physique". Princeton Univ. Bull, vol. 13, pp. 49-52, 1902.

[12] P. Charbonnier,L. Blanc-Feraud, G. Aubert, M. Barlaud. Deterministic edge-preserving regularization in computed imaging. IEEE Trans Image Process, pp. 298-311, 1997.

[13] A. Tikhonov. "Tikhonov regularization of incorrectly posed problems". Soviet Mathematics Doklady, vol. 4, pp. 1624-1627, 1963.

[14] Xiaojuan Gu, Li Gao. A new method for parameter estimation of edge-preserving regularization in image restoration. Journal of Computational and Applied Mathematics, vol. 225, iss. 2, pp. 478-486, 2009.

[15] Wenyu Sun, Ya-xiang Yuan. Optimization Theory and Methods. Nonlinear Programming, Springer, 2006.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick. A multigrid tutorial. SIAM, 2000.

[17] L. Jacques, L. Duval, C. Chaux, G. Peyré, Gabriel. A Panorama on Multiscale Geometric Representations, Intertwining Spatial, Directional and Frequency Selectivity. Signal Processing, 91, 2018.

[18] I. W. Selesnick, M. A. T. Figueiredo. Signal restoration with overcomplete wavelet transforms: Comparison of analysis and synthesis priors. Wavelets XIII, vol. 7446, pp. 107-121, 2009.

[19] P. Parpas. A Multilevel Proximal Gradient Algorithm for a Class of Composite Optimization Problems. SIAM Journal on Scientific Computing, vol. 39. pp. 681-701. 2017.

[20] L. Rudin, S. Osher, E. Fatemi. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena. vol. 60, pp. 259-268, 1992.

[21] J. Nocedal and S. Wright. Numerical optimization. Springer Science and Business Media, 2006.