Università degli Studi di Genova



Università
di Genova

Dipartimento di Matematica

Corso di Laurea Magistrale in

Matematica

Prova finale

Proximal Gauss–Newton method in a multilevel framework

Candidata:

Camilla Maria Graziani

Relatrici:
Prof.ssa Silvia Villa
Prof.ssa Elisa Riccietti

Correlatore:
Prof. Luca Calatroni

Anno Accademico 2025/2026

Table of Contents

Introduction **3**

1 Preliminaries **5**
1.1 Optimization 5
1.2 Algorithms 8
1.3 Backtracking strategy 12

2 Considered methods **14**
2.1 Proximal Gradient Descent method 15
2.2 Proximal Gauss–Newton method 15
2.2.1 Numerical comparison between PGD and PGN 18
2.3 The Levenberg–Marquardt method 19
2.3.1 The Levenberg–Marquardt approach for the PGN method 21
2.3.2 Convergence analysis 22

3 Multilevel framework **27**
3.1 Multilevel optimization in the smooth setting 28
3.2 Coarse-level construction 29
3.3 Non-smooth multilevel optimization 34
3.3.1 Smooth coarse model 35
3.3.2 Fine-level decrease 37
3.3.3 Non-smooth coarse model 39
3.3.4 Fine-level decrease for non-smooth coarse models 39
3.4 Multilevel Proximal Gauss–Newton method 41

4 Numerical implementation and results **44**
4.1 Smooth coarse problem 44
4.1.1 1D case 45
4.1.2 Numerical experiments in 1D 49
4.1.3 2D problem 53
4.1.4 Numerical experiments in 2D 55
4.2 Non-smooth coarse problem 56
4.2.1 Construction of the coarse constraint set 57

4.2.2 Setting with an active box constraint 58
4.2.3 Numerical results in 1D 59
4.2.4 Numerical results in 2D 64

Conclusions & Outlook **69**

Bibliography **71**

Introduction

Inverse problems and machine learning play a fundamentally important role in the field of applied mathematics. In most cases, addressing these tasks intrinsically translates into formulating and solving an optimization problem. Contemporary challenges in these fields are characterized by a massive dimensional scale; applications such as high-resolution image reconstruction, the training of deep neural networks, and complex signal processing generate large-scale problems that require managing millions of parameters, rendering classical solvers impractical.

A significant portion of these problems can be cast as composite optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \Phi(x) := \frac{1}{2}\|F(x) - y\|^2 + J(x), \tag{P}$$

where F is a, possibly nonlinear, operator modeling the direct problem, and J is a convex, possibly non-smooth, functional encoding a regularization term or a feasibility constraint. Currently, the most widely adopted approach to solving problems of the form (P) relies on first-order methods, most notably Proximal Gradient Descent. While each iteration is computationally cheap, these methods typically exhibit slow, sublinear convergence, a limitation that becomes especially significant for large-scale problems.

In smooth optimization, the advantages of second-order methods over first-order schemes are well established: by exploiting curvature information, methods such as Newton or Gauss–Newton achieve fast local convergence and require far fewer iterations. The *Proximal Gauss–Newton (PGN) method* extends this idea to the composite setting (P). However, PGN can be computationally demanding: at each iteration, it requires forming the Gauss–Newton approximation of the Hessian, $F'(x_k)^\top F'(x_k)$, solving the associated linear system, and computing a proximity operator in the metric induced by this matrix. As the problem dimension grows, these operations become increasingly expensive.

Multilevel (ML) frameworks offer a natural way to reduce this cost, by exploiting coarser representations of the problem to compute inexpensive correction steps. In smooth optimization, combining second-order methods with multilevel techniques has proved very effective. In the non-smooth setting, instead, existing multilevel approaches are mainly based on first-order methods; however, by considering multilevel variants of second-order methods, one can further speed

up the solution, a direction that has received considerably less attention. In this thesis, we take a step in this direction. We study the PGN method and, to handle cases in which the Jacobian of F becomes ill-conditioned or singular, we introduce a Levenberg–Marquardt regularization of the method and establish its local convergence. We then combine these methods with a multilevel framework and investigate numerically the effect of the regularization and of the coarse corrections on a nonlinear elliptic boundary value problem subject to box constraints, in both one and two spatial dimensions.

The remainder of this thesis is organized as follows:

- **Chapter 1** reviews the key mathematical definitions and preliminary results necessary for our study.
- **Chapter 2** presents the formulation of the target problem and introduces the fundamental iterative techniques to solve it, namely Proximal Gradient Descent (PGD) and the Proximal Gauss–Newton (PGN) method, highlighting their structural differences. We then propose a Levenberg–Marquardt regularization of PGN (PGN-LM), and establish its local convergence.
- **Chapter 3** introduces the multilevel optimization framework, first for smooth objectives and then in the non-smooth composite setting. Building on this, we propose multilevel versions of the Proximal Gauss–Newton method and of its Levenberg–Marquardt variant.
- **Chapter 4** details our numerical implementation and discusses the experimental results obtained on the one- and two-dimensional problems, providing a comparative analysis of the algorithms' efficiency.
- **Conclusions & Outlook** summarizes the main contributions of this work and outlines possible directions for future research.

Chapter 1

Preliminaries

In this chapter, we introduce the main mathematical concepts and definitions that will be used throughout the thesis. Most of the material presented in this chapter is based on [1].

Throughout this thesis, unless otherwise specified, we work in $\mathbb{R}^n$, endowed with the standard inner product and the associated Euclidean norm, denoted by

$$\langle \cdot, \cdot \rangle \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \qquad \text{and} \qquad \| \cdot \| \colon \mathbb{R}^n \to \mathbb{R},$$

respectively.

For every $x \in \mathbb{R}^n$ and every $r > 0$, we denote by $B_r(x) := \{y \in \mathbb{R}^n : \|y - x\| < r\}$ the open ball centered at x with radius r.

We define the distance from a point $x_0 \in \mathbb{R}^n$ to a set $S \subseteq \mathbb{R}^n$ as the infimum of the distances between x_0 and all points in S, that is,

$$\operatorname{dist}(x_0, S) = \inf_{y \in S} \|x_0 - y\|.$$

Definition 1.1 (Star-shaped set)**.** *A subset $S \subseteq \mathbb{R}^n$ is called* star-shaped *(or radially convex) with respect to a point $x_0 \in S$ if the line segment joining x_0 and any point $x \in S$ is contained in S:*

$$[x_0, x] := \{(1-t)x_0 + tx \mid t \in [0,1]\} \subseteq S \qquad \forall x \in S. \tag{1.1}$$

1.1 Optimization

Definition 1.2 (Effective domain)**.** *Let $f \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be an extended real-valued function. The* effective domain *of f, denoted by $\mathbf{dom}(f)$, is defined as the set of points where the function takes finite values:*

$$\mathbf{dom}(f) := \{x \in \mathbb{R}^n \mid f(x) < +\infty\}. \tag{1.2}$$

Definition 1.3 (Convex function)**.** *A function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is* convex *if, for all $x, y \in \mathbb{R}^n$ and all $t \in [0,1]$,*

$$f((1-t)x + ty) \leq (1-t)f(x) + tf(y). \tag{1.3}$$

The function f is strictly convex *if*

$$f((1-t)x + ty) < (1-t)f(x) + tf(y)$$

for all $x, y \in \mathbf{dom} f$ with $x \neq y$ and all $t \in (0,1)$.

Definition 1.4 (Proper function)**.** *A function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to be* proper *if its effective domain is nonempty. Equivalently, f is proper if there exists at least one $u \in \mathbb{R}^n$ such that $f(u) < +\infty$.*

Definition 1.5 (Lower semicontinuous function)**.** *A function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is* lower semicontinuous *if for every sequence $(x_k)_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ and every $x \in \mathbb{R}^n$,*

$$x_k \to x \implies f(x) \leq \liminf_{k\to\infty} f(x_k).$$

Definition 1.6. *We denote by $\Gamma_0(\mathbb{R}^n)$ the class of functions $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ that are proper, convex, and lower semicontinuous.*

Definition 1.7 (Convex set)**.** *A subset $C \subseteq \mathbb{R}^n$ is called* convex *if for all $x, y \in C$ and for all $t \in [0,1]$, it holds that*

$$(1-t)x + ty \in C. \tag{1.4}$$

Among the most common regularity conditions in optimization is L-smoothness, which restricts how rapidly the gradient of a function can change.

Definition 1.8 (L-smoothness)**.** *A continuously differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$ is said to be L-*smooth *if its gradient ∇f is L-Lipschitz continuous, i.e., there exists a constant $L > 0$ such that*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \qquad \forall x, y \in \mathbb{R}^n. \tag{1.5}$$

For convex functions that are not differentiable, the gradient is replaced by a fundamental tool in non-smooth analysis, known as the subdifferential.

Definition 1.9 (Subdifferential)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper convex function and fix $x \in \mathbf{dom}(f)$. The* subdifferential *of f at x is defined as*

$$\partial f(x) := \{u \in \mathbb{R}^n \mid f(y) \geq f(x) + \langle y - x, u\rangle \quad \forall y \in \mathbb{R}^n\}.$$

Each element of $\partial f(x)$ is called a subgradient *of f at x. If $x \notin \mathbf{dom}(f)$, we set $\partial f(x) = \emptyset$.*

Theorem 1.10 (Fermat's theorem)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper convex function. Then the following conditions are equivalent:*

(a) $x_* \in \mathbb{R}^n$ is a global minimizer of f;

(b) $0 \in \partial f(x_*)$.

A fundamental tool for handling non-smooth convex functions is the proximity operator. As we will see, it plays a central role in proximal algorithms.

Definition 1.11 (Proximity operator)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper, convex and lower semicontinuous (lsc) function. The* proximity operator *of f is defined as*

$$\mathrm{prox}_f\colon \mathbb{R}^n \to \mathbb{R}^n, \qquad \mathrm{prox}_f(x) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(f(u) + \frac{1}{2}\|x - u\|^2 \right).$$

The proximity operator is well defined: since f is proper, convex and lsc, the function $u \mapsto f(u) + \frac{1}{2}\|x - u\|^2$ is proper, lsc and strongly convex, hence it admits a unique minimizer for every $x \in \mathbb{R}^n$.

Example 1.12 (Indicator function and projection operator)**.** *Let $C \subseteq \mathbb{R}^n$ be a nonempty, closed and convex set. The indicator function of C is defined as*

$$\iota_C(x) = \begin{cases} 0, & \text{if } x \in C, \\ +\infty, & \text{if } x \notin C. \end{cases}$$

For every $\alpha > 0$, the proximity operator of $\alpha\, \iota_C$ is the orthogonal projection onto C, denoted by P_C:

$$\mathrm{prox}_{\alpha\iota_C}(x) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \left(\iota_C(z) + \frac{1}{2\alpha}\|z - x\|^2 \right) = \operatorname*{argmin}_{z \in C} \ \frac{1}{2}\|z - x\|^2 = P_C(x).$$

An important special case is that of box constraints. Given vectors $a, b \in \mathbb{R}^n$ with $a_i \leq b_i$ for $i = 1, \ldots, n$, consider the box

$$C = [a, b] = \{x \in \mathbb{R}^n : a_i \leq x_i \leq b_i,\ i = 1, \ldots, n\}.$$

In this case, the projection can be computed explicitly, component by component:

$$\big(P_C(x)\big)_i = \begin{cases} a_i, & \text{if } x_i < a_i, \\ x_i, & \text{if } a_i \le x_i \le b_i, \\ b_i, & \text{if } x_i > b_i, \end{cases} \qquad i = 1, \dots, n.$$

We recall a fundamental property of the proximity operator, namely its firm non-expansiveness.

Proposition 1.13 (Firm non-expansiveness of the proximity operator)**.** *Let $g \in \Gamma_0(\mathbb{R}^n)$. Then, for all $x, y \in \mathbb{R}^n$,*

$$\| \operatorname{prox}_g(x) - \operatorname{prox}_g(y) \|^2 \le \langle x - y, \operatorname{prox}_g(x) - \operatorname{prox}_g(y) \rangle.$$

Remark 1.14. *This property is not merely a technical formality: it is the cornerstone for proving the convergence of non-smooth optimization algorithms. To understand its significance, it is useful to compare it with other standard properties of an operator $T\colon \mathbb{R}^n \to \mathbb{R}^n$:*

- ***Strict contraction*** *($\|T(x) - T(y)\| \le c\|x - y\|$ with $c < 1$): it strictly reduces distances, but the proximity operator is in general* not *a strict contraction.*
- ***Non-expansiveness*** *($\|T(x) - T(y)\| \le \|x - y\|$): it ensures that distances do not increase, but alone it cannot guarantee convergence to a fixed point (as it allows distance-preserving rotations).*
- ***Firm non-expansiveness****: a strictly stronger condition than non-expansiveness. It provides an additional dissipation property, which allows fixed-point iterations to converge.*

1.2 Algorithms

We now recall the main optimization algorithms that will be used throughout this manuscript.

Gradient Descent (GD)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable and convex function. Consider the minimization problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^n} \; f(x). \tag{1.6}$$

Problem (1.6) can be solved using *gradient descent*, which, given $\alpha > 0$ and $x_0 \in \mathbb{R}^n$, generates the sequence

$$x_{k+1} = x_k - \alpha \nabla f(x_k). \tag{1.7}$$

If f is L-smooth, $\alpha \in \left(0, \frac{2}{L}\right)$, and $\operatorname{argmin}(f) \neq \emptyset$, then the sequence $(x_k)_{k\in\mathbb{N}}$ generated by (1.7) converges to a global minimizer x_* of f.

Theorem 1.15 (Convergence of gradient descent)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable, with L-Lipschitz continuous gradient. Assume that $S := \operatorname{argmin} f \neq \emptyset$ and set $f_* := \min f$. Let $x_0 \in \mathbb{R}^n$, $0 < \alpha < 2/L$, and let $(x_k)_{k\in\mathbb{N}}$ be generated by (1.7). Then the following statements hold.*

(i) $\displaystyle\sum_{k\in\mathbb{N}} \|x_{k+1} - x_k\|^2 \leq \frac{2}{2-\alpha L} \operatorname{dist}(x_0, S)^2.$

(ii) For every $k \in \mathbb{N}$ and every $x \in \mathbb{R}^n$,

$$\|x_{k+1} - x\|^2 \leq \|x_k - x\|^2 + 2\alpha\big(f(x) - f(x_{k+1})\big) + (\alpha L - 1)\|x_{k+1} - x_k\|^2.$$

(iii) For every $k \in \mathbb{N}$,

$$\Big(\frac{1}{\alpha} - \frac{L}{2}\Big)\|x_{k+1} - x_k\|^2 \leq f(x_k) - f(x_{k+1}),$$

so that the algorithm is descending.

(iv) $f(x_{k+1}) - f_* = o\big(1/(k+1)\big)$ and, for every $k \in \mathbb{N}$,

$$f(x_{k+1}) - f_* \leq \frac{\operatorname{dist}(x_0, S)^2}{k+1} \times \begin{cases} \dfrac{1}{2\alpha} & \text{if } \alpha \leq \dfrac{1}{L}, \\ \dfrac{L}{2}\,\dfrac{1}{2-\alpha L} & \text{if } \dfrac{1}{L} < \alpha < \dfrac{2}{L}. \end{cases}$$

(v) The sequence $(x_k)_{k\in\mathbb{N}}$ converges to some $x_* \in S$.

Gradient descent relies on smoothness and differentiability assumptions; we now introduce proximal algorithms, which circumvent these hypotheses and remain applicable to non-smooth objectives.

Proximal Gradient Descent (PGD)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function, and let $g\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper, lower semicontinuous, convex function. Consider the problem

$$\operatorname*{argmin}_{x\in\mathbb{R}^n} \; f(x) + g(x). \tag{1.8}$$

Proximal gradient descent allows us to solve (1.8) through the following scheme, given $\alpha > 0$ and $x_0 \in \mathbb{R}^n$:

$$x_{k+1} = \operatorname{prox}_{\alpha g}\big(x_k - \alpha \nabla f(x_k)\big). \tag{1.9}$$

If f is L-smooth, $\alpha \in \left(0, \frac{2}{L}\right)$, and $\operatorname{argmin}(f+g) \neq \emptyset$, then the sequence $(x_k)_{k\in\mathbb{N}}$ generated by (1.9) converges to a global minimizer x_* of $f+g$.

Example 1.16 (Projected gradient method)**.** *Let $g = \iota_C$ be the indicator function of a nonempty, closed and convex set $C \subseteq \mathbb{R}^n$. As shown in Example 1.12, its proximity operator is the projection onto C. In this case, the Proximal Gradient scheme* (1.9) *reduces to the* projected gradient method

$$x_{k+1} = P_C\big(x_k - \alpha \nabla f(x_k)\big), \tag{1.10}$$

which alternates a gradient step on f with a projection onto the feasible set.

Theorem 1.17 (Convergence of Proximal Gradient Descent [1])**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable, with L-Lipschitz continuous gradient, and let $g \in \Gamma_0(\mathbb{R}^n)$. Set $\Phi := f + g$, assume that $S := \operatorname{argmin}\Phi \neq \emptyset$ and set $\Phi_* := \min\Phi$. Let $x_0 \in \mathbb{R}^n$, $0 < \alpha < 2/L$, and let $(x_k)_{k\in\mathbb{N}}$ be generated by* (1.9)*. Then the following statements hold.*

(i) $\displaystyle\sum_{k\in\mathbb{N}} \|x_{k+1} - x_k\|^2 \leq \frac{2}{2-\alpha L}\,\operatorname{dist}(x_0, S)^2.$

(ii) For every $k \in \mathbb{N}$ and every $x \in \mathbb{R}^n$,

$$\|x_{k+1} - x\|^2 \leq \|x_k - x\|^2 + 2\alpha\big(\Phi(x) - \Phi(x_{k+1})\big) + (\alpha L - 1)\,\|x_{k+1} - x_k\|^2.$$

(iii) For every $k \in \mathbb{N}$,

$$\Big(\frac{1}{\alpha} - \frac{L}{2}\Big)\|x_{k+1} - x_k\|^2 \leq \Phi(x_k) - \Phi(x_{k+1}),$$

so that the algorithm is descending.

(iv) $\Phi(x_{k+1}) - \Phi_* = o\big(1/(k+1)\big)$ *and, for every $k \in \mathbb{N}$,*

$$\Phi(x_{k+1}) - \Phi_* \leq \frac{\operatorname{dist}(x_0, S)^2}{k+1} \times \begin{cases} \dfrac{1}{2\alpha} & \text{if } \alpha \leq \dfrac{1}{L}, \\ \dfrac{L}{2}\,\dfrac{1}{2-\alpha L} & \text{if } \dfrac{1}{L} < \alpha < \dfrac{2}{L}. \end{cases}$$

(v) The sequence $(x_k)_{k\in\mathbb{N}}$ converges to some $x_* \in S$.

A limitation of PGD can be its slow convergence. The convergence rate stated in the fourth item makes this limitation precise: in the worst case, the objective error $\Phi(x_{k+1}) - \Phi_*$ decreases only as $\mathcal{O}(1/k)$, which is a sublinear rate. This is an upper bound valid for every problem in the considered class; on specific problems the method may converge faster, but no better rate can be guaranteed in general.

Convergence results are also available in the nonconvex setting. Under suitable regularity assumptions on the objective, boundedness of the iterates, and an appropriate choice of the step size, Proximal Gradient methods can be shown to converge to a critical point [2]. Such a point, however, need not be a local or global minimizer.

Gauss–Newton (GN)

Let $F\colon \mathbb{R}^n \to \mathbb{R}^m$ (with $m \geq n$) be a continuously differentiable function. Consider the nonlinear least-squares problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^n} \ \frac{1}{2}\|F(x)\|^2. \tag{1.11}$$

The Gauss–Newton algorithm allows us to solve (1.11) through the following scheme, given an initial guess $x_0 \in \mathbb{R}^n$ and letting $F'(x)$ denote the $m \times n$ Jacobian matrix of F evaluated at x:

$$x_{k+1} = x_k - \big(F'(x_k)^\top F'(x_k)\big)^{-1} F'(x_k)^\top F(x_k). \tag{1.12}$$

The method is well defined as long as $F'(x_k)^\top F'(x_k)$ is invertible, that is, $F'(x_k)$ has full column rank. Its local convergence is described by the following result.

Theorem 1.18 ([3, Theorem 10.2.1])**.** *Let $F\colon \mathbb{R}^n \to \mathbb{R}^m$ and let $f(x) = \frac{1}{2}\|F(x)\|^2$ be twice continuously differentiable in an open convex set $D \subseteq \mathbb{R}^n$. Assume that F' is Lipschitz continuous on D with constant γ, and that $\|F'(x)\| \leq \beta$ for all $x \in D$. Assume moreover that there exist $x_* \in D$ and $\lambda, \sigma \geq 0$ such that $F'(x_*)^\top F(x_*) = 0$, λ is the smallest eigenvalue of $F'(x_*)^\top F'(x_*)$, and*

$$\big\|\big(F'(x) - F'(x_*)\big)^\top F(x_*)\big\| \leq \sigma\|x - x_*\| \qquad \text{for all } x \in D. \tag{1.13}$$

If $\sigma < \lambda$, then for any $c \in (1, \lambda/\sigma)$ there exists $\varepsilon > 0$ such that, for all $x_0 \in B_\varepsilon(x_*)$, the sequence generated by (1.12) is well defined, converges to x_*, and satisfies

$$\|x_{k+1} - x_*\| \leq \frac{c\,\sigma}{\lambda}\,\|x_k - x_*\| + \frac{c\,\beta\gamma}{2\lambda}\,\|x_k - x_*\|^2 \tag{1.14}$$

and

$$\|x_{k+1} - x_*\| \leq \frac{c\,\sigma + \lambda}{2\lambda}\,\|x_k - x_*\| < \|x_k - x_*\|.$$

The constant σ measures the combined effect of the nonlinearity of F and of the size of the residual at x_*: in particular, if $F(x_*) = 0$, condition (1.13) holds with $\sigma = 0$, and (1.14) shows that the convergence is quadratic. If instead the residual is nonzero, the first term in (1.14) dominates and the convergence is only linear, with a rate that deteriorates as σ approaches λ; for $\sigma \geq \lambda$, convergence is no longer guaranteed.

1.3 Backtracking strategy

Many iterative optimization methods can be written in the general form

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.15}$$

where d_k is a *search direction*, which specifies along which direction the current iterate is moved, and $\alpha_k > 0$ is a *step length*, which specifies how far to move along it. Typically, d_k is chosen as a *descent direction* for the objective f at x_k, that is,

$$\langle \nabla f(x_k), d_k \rangle < 0,$$

so that f decreases along d_k, at least for sufficiently small steps.

Once a search direction d_k has been computed, the remaining task is to select an admissible step length α_k. The fact that d_k is a descent direction only ensures that f decreases for small enough steps: a step that is too long may even increase f, while a step that is too short leads to slow progress. Moreover, requiring only $f(x_k + \alpha_k d_k) < f(x_k)$ is not enough in general, since the decrease may become arbitrarily small and the iterates may stall at a non-stationary point. A standard way to avoid this is to require a *sufficient decrease*, expressed by the *Armijo condition*

$$f(x_k + \alpha d_k) \leq f(x_k) + c\,\alpha\,\langle \nabla f(x_k), d_k \rangle, \tag{1.16}$$

with $c \in (0, 1)$, typically small (e.g. $c = 10^{-4}$). Since d_k is a descent direction, the last term is negative, so that (1.16) requires the actual decrease of f to be at least a fraction c of the decrease predicted by the first-order approximation of f along d_k. For a descent direction, this condition is always satisfied for sufficiently small α.

In this work, we adopt a backtracking strategy, which is simple and computationally inexpensive: starting from a unit step, α is reduced by a constant factor $\tau \in (0, 1)$ until (1.16) is satisfied. A sketch of the procedure is given in Algorithm 1.

Algorithm 1 Backtracking line search (Armijo condition)

Require: Objective $f\colon \mathbb{R}^n \to \mathbb{R}$, current point x_k, descent direction d_k, parameters $c \in (0,1)$, $\tau \in (0,1)$, maximum number of reductions $M_{\max} \in \mathbb{N}$.
Ensure: Next iterate x_{k+1}.

Set $\alpha = 1$ and $m = 0$
while $m < M_{\max}$ **do**
    if condition (1.16) holds **then**
        return $x_{k+1} = x_k + \alpha d_k$
    end if
    $\alpha \leftarrow \tau\alpha, \quad m \leftarrow m + 1$
end while
return $x_{k+1} = x_k$ ▷ safeguard: no sufficient decrease found

In the numerical experiments, since the composite objectives considered in the following chapters are not differentiable, we use a simplified version of the Armijo condition, obtained by taking $c = 0$, which reduces to the simple decrease condition $f(x_k + \alpha d_k) < f(x_k)$.

Chapter 2

Considered methods

Our main problem of interest consists in solving the following composite optimization problem. Given a differentiable nonlinear map $F\colon \mathbb{R}^n \to \mathbb{R}^m$, a data vector $y \in \mathbb{R}^m$, and a proper, convex, lower semicontinuous penalty function $J\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, we consider the optimization problem

$$\min_{x \in \mathbb{R}^n} \ \Phi(x) := \frac{1}{2}\|F(x) - y\|^2 + J(x). \tag{P}$$

The first term in (P) is a nonlinear least-squares function, which measures the discrepancy between the model output $F(x)$ and the data y; problem (P) can therefore be seen as a regularized, or constrained, nonlinear least-squares problem. In general, it is nonconvex and non-smooth; it does, however, possess a particular structure, being the sum of a nonconvex, smooth term (the data-fidelity term) and a convex, possibly non-smooth one (the regularization term).

We point out that the above assumptions do not ensure that problem (P) admits a solution: since F is nonlinear, Φ is in general neither convex nor coercive. For this reason, the existence of a (local) minimizer of Φ will be explicitly assumed in each result where it is needed.

Problems of the form (P) arise naturally in a wide variety of applied fields:

- **Image processing:** tasks such as denoising, deblurring, and MRI/CT reconstruction.
- **Partial differential equations (PDEs):** finding the solution of a nonlinear PDE, typically after discretization.
- **Machine learning:** training of neural networks.

Currently, to tackle problems of this form, the scientific community predominantly relies on first-order optimization methods, most notably the Proximal Gradient Descent (PGD) method.

2.1 Proximal Gradient Descent method

Without loss of generality, we can assume $y = 0$, since the general case is recovered by replacing F with $F - y$, which has the same Jacobian. In the following, we therefore consider

$$\min_{x \in \mathbb{R}^n} \Phi(x) = G(x) + J(x), \qquad G(x) := \frac{1}{2}\|F(x)\|^2. \tag{2.1}$$

At each iteration k, PGD updates the current iterate x_k by performing a gradient descent step with respect to the smooth part G, followed by a proximal step with respect to the non-smooth function J. The iteration is given by

$$x_{k+1} = \operatorname{prox}_{\alpha J}\big(x_k - \alpha \nabla G(x_k)\big), \tag{2.2}$$

where $\alpha > 0$ is the step size. The argument of the proximity operator, $x_k - \alpha \nabla G(x_k)$, corresponds exactly to a gradient descent step on G, whose gradient is

$$\nabla G(x_k) = F'(x_k)^\top F(x_k).$$

The map $\operatorname{prox}_{\alpha J} \colon \mathbb{R}^n \to \mathbb{R}^n$ is the *proximity operator* of αJ, defined for every $v \in \mathbb{R}^n$ as

$$\operatorname{prox}_{\alpha J}(v) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \left\{ J(z) + \frac{1}{2\alpha}\|z - v\|^2 \right\}. \tag{2.3}$$

The success of these first-order methods stems from the extremely low computational cost of each iteration. However, they exhibit some crucial limitations, in particular slow convergence: as stated in Theorem 1.17, in the worst case the objective error decreases only at a sublinear rate. To overcome this limitation, an effective approach is to resort to second-order optimization methods, such as the well-known Newton method. By exploiting the second derivatives of the smooth term, encoded in the Hessian matrix $\nabla^2 G(x)$, these methods capture the local curvature of the objective function. This additional information allows for significantly faster convergence, typically quadratic in a neighborhood of the solution, at the price of a higher cost per iteration. To this end, we consider the Proximal Gauss–Newton (PGN) method [4], where $F \colon \mathbb{R}^n \to \mathbb{R}^m$ is a smooth nonlinear residual function and $J \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper, lower semicontinuous, and convex (possibly non-smooth) penalty function.

2.2 Proximal Gauss–Newton method

Proximal Gauss–Newton is inherently a second-order method, but it utilizes an approximation of the true Hessian matrix $\nabla^2 G(x)$ to significantly reduce computational costs. Assuming $F : \mathbb{R}^n \to \mathbb{R}^m$ is twice continuously differentiable, the Hessian of the nonlinear least-squares functional is given by

$$\nabla^2 G(x_k) = F'(x_k)^\top F'(x_k) + \sum_{i=1}^{m} F_i(x_k) \nabla^2 F_i(x_k).$$

The Proximal Gauss–Newton method neglects the second-order terms of the residual, using the approximation

$$\nabla^2 G(x_k) \approx F'(x_k)^\top F'(x_k) =: H(x_k).$$

Using this approximation, the PGN iteration is given by

$$x_{k+1} = \operatorname{prox}_J^{H(x_k)} \left(x_k - H(x_k)^{-1} F'(x_k)^\top F(x_k) \right). \tag{2.4}$$

The term inside the parentheses is the classical Gauss–Newton step, introduced in Section 1.2 and extensively studied in [3]. The operator $\operatorname{prox}_J^{H(x_k)}$ is the proximity operator associated with the non-smooth term J, evaluated with respect to the metric induced by $H(x_k)$, and is defined as

$$\operatorname{prox}_J^{H(x_k)}(v) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \left\{ J(z) + \frac{1}{2} \|z - v\|^2_{H(x_k)} \right\},$$

where the scaled norm can be expressed as

$$\|z - v\|^2_{H(x_k)} = \langle F'(x_k)^\top F'(x_k)(z - v), z - v \rangle = \|F'(x_k)(z - v)\|^2.$$

The Proximal Gauss–Newton method boasts several highly desirable theoretical properties. Most notably, it ensures fast local convergence rates (often superlinear or even quadratic in favorable scenarios) while bypassing the need to compute the exact, computationally prohibitive second-order derivatives of the residual F. In the following theorem, we recall a local convergence analysis of the Proximal Gauss–Newton method.

Theorem 2.1 (Local convergence of PGN [4])**.** *Let $\Phi(x) \coloneqq \frac{1}{2}\|F(x)\|^2 + J(x)$, where*

- *$F \colon \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ is differentiable, with Ω open;*
- *$J \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is proper, lower semicontinuous and convex.*

Let $x_* \in \operatorname{dom}(J)$ be a local minimizer of Φ, and let $U \subseteq \Omega$ be an open set, star-shaped with respect to x_*. Let $R > 0$ be such that $B_R(x_*) \subseteq U$. Moreover, assume that:

1. *$F'(x_*) \in \mathbb{R}^{m \times n}$ has full column rank, i.e., it is injective;*
2. *$F' \colon U \to \mathbb{R}^{m \times n}$ is Lipschitz continuous with constant $L > 0$;*
3. *$\left[(1 + \sqrt{2})\kappa + 1\right] \theta \beta^2 L < 1$, where*

$$\theta = \|F(x_*)\|, \qquad \beta = \left\| [F'(x_*)^\top F'(x_*)]^{-1} F'(x_*)^\top \right\|, \qquad \kappa = \beta \, \|F'(x_*)\|,$$

the latter being the condition number of $F'(x_*)$.

Define $\bar{R} = \min\{R, 1/(\beta L)\}$ *and* $q\colon [0, \bar{R}) \to \mathbb{R}_+$ *as*

$$q(r) = \frac{\beta}{(1-\beta L r)^2}\Big[\tfrac{3}{2}\big(\beta L^2 r^2 + \kappa L r\big) + (1+\sqrt{2})\,\theta\beta^2 L^2 r + \big[(1+\sqrt{2})\kappa + 1\big]\theta\beta L\Big].$$

The function q *is continuous and strictly increasing. Define*

$$\bar{r} = \sup\{r \in (0, \bar{R}) : q(r) < 1\}, \tag{2.5}$$

and fix $r \in \mathbb{R}$*, with* $0 < r \leq \bar{r}$*. Then, for every* $x_0 \in B_r(x_*)$*, the sequence generated by*

$$x_{k+1} = \operatorname{prox}_J^{H(x_k)}\big(x_k - H(x_k)^{-1}F'(x_k)^\top F(x_k)\big), \qquad H(x_k) \coloneqq F'(x_k)^\top F'(x_k),$$

is well defined, i.e. $x_k \in B_r(x_*)$ *and* $F'(x_k)$ *has full column rank for every* k*, and it holds*

$$\|x_k - x_*\| \leq q_0^k \|x_0 - x_*\|, \qquad q_0 \coloneqq q(\|x_0 - x_*\|) < 1.$$

More precisely,

$$\|x_{k+1} - x_*\| \leq C_2\|x_k - x_*\|^2 + C_1\|x_k - x_*\|,$$

with

$$C_1 = \frac{\big[(1+\sqrt{2})\kappa + 1\big]\theta\beta^2 L}{(1-\beta L\rho_{x_0})^2},$$

$$C_2 = \beta\,\frac{3\kappa L + 2(1+\sqrt{2})\theta\beta^2 L^2 + 3\beta L^2\rho_{x_0}}{2(1-\beta L\rho_{x_0})^2},$$

where $\rho_{x_0} = \|x_0 - x_*\|$*.*

Since $\bar{r}$ is the largest radius for which $q(r) < 1$, the ball $B_{\bar{r}}(x_*)$ is a region of guaranteed convergence around the local minimizer x_*. The second estimate of the theorem also describes the type of convergence of PGN. The constant C_1 is proportional to the residual $\theta = \|F(x_*)\|$. If $\theta > 0$, then $C_1 > 0$, and close to the solution the linear term dominates, so that the convergence is in general only linear. If instead the residual is zero, $\theta = 0$, then $C_1 = 0$ and the estimate reduces to $\|x_{k+1} - x_*\| \leq C_2\|x_k - x_*\|^2$, that is, the convergence is quadratic, as for the classical Gauss–Newton method on zero-residual problems. To illustrate the different behavior of first- and second-order methods, we now present a numerical example comparing PGD and PGN on a common test problem.

2.2.1 Numerical comparison between PGD and PGN

We now present a numerical example comparing PGN and PGD on a common test problem. We consider the generalized Rosenbrock function in n variables, formulated as a nonlinear least-squares problem. Specifically, we define the residual vector $F : \mathbb{R}^n \to \mathbb{R}^{2(n-1)}$ component-wise, for $i = 1, \dots, n-1$, as

$$F_{2i-1}(x) = 10\left(x_{i+1} - x_i^2\right), \qquad F_{2i}(x) = 1 - x_i,$$

so that the corresponding least-squares term

$$G(x) = \frac{1}{2}\|F(x)\|^2 = 50\sum_{i=1}^{n-1}\left(x_{i+1} - x_i^2\right)^2 + \frac{1}{2}\sum_{i=1}^{n-1}(1 - x_i)^2$$

is, up to a factor $\frac{1}{2}$, the generalized Rosenbrock function, whose global minimizer is $x_* = (1, \dots, 1) \in \mathbb{R}^n$, with $G(x_*) = 0$. We add the box constraint $C = [0, 2]^n$ and consider the composite problem

$$\min_{x \in \mathbb{R}^n} \; G(x) + J(x), \qquad J = \iota_C.$$

Since $x_* \in C$, it is also the solution of the constrained problem. We consider $n \in \{100, 300, 500\}$ and the initial guess $x_0 = 0.5\,(1, \dots, 1)$. Both methods are used with a fixed step.

- *PGD.* We use the constant step size $\alpha = 10^{-4}$, chosen by trial and error as the largest value for which the method behaved stably.
- *PGN.* The scaled projection $P_C^{H(x_k)}(w_k)$ does not admit a closed form, since $H(x_k)$ is not diagonal. Thus it is computed by iteratively solving the minimization problem associated with the proximity operator; in particular, as discussed in detail in Chapter 4, it is approximated by the projected gradient iterations (4.8), with step $\sigma_k = 1/(2\lambda_{\max}(H(x_k)))$.

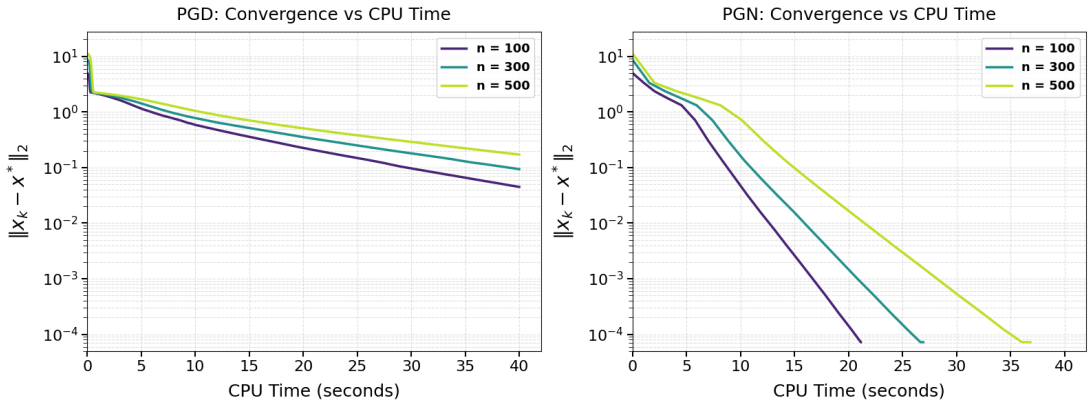


Figure 2.1: Convergence of PGD and PGN in terms of CPU time for different problem dimensions n.

Figure 2.1 compares Proximal Gradient Descent (PGD) and Proximal Gauss–Newton (PGN), plotting the distance to the optimum $\|x_k - x_*\|_2$ against CPU time for problem dimensions $n = 100$, 300 and 500. After a rapid initial decrease, PGD converges very slowly: within 40 seconds, the error only reaches values between about $5 \cdot 10^{-2}$ and $2 \cdot 10^{-1}$, depending on the dimension. PGN, in contrast, shows a much steeper decrease and reaches an accuracy below 10^{-4} in about 21, 27 and 37 seconds for $n = 100$, 300 and 500, respectively. In both cases, the computational time required to reach a given accuracy increases with the problem dimension, since the cost of each iteration grows with n.

One limitation of the Proximal Gauss–Newton method, however, is that it requires $F'(x_k)$ to be injective at every iteration, so that $H(x_k) = F'(x_k)^\top F'(x_k)$ is invertible, an assumption that may fail, or become numerically unstable, in practice. In the following section, we present a strategy to circumvent this limitation by introducing the Levenberg–Marquardt method, which is an iterative procedure for the solution of least-squares problems.

2.3 The Levenberg–Marquardt method

The Levenberg–Marquardt (LM) algorithm is a widely used optimization technique specifically designed to solve nonlinear least-squares problems. Recall that the classical nonlinear least-squares problem consists in minimizing

$$f(x) = \frac{1}{2}\|F(x)\|^2, \tag{2.6}$$

with $F\colon D \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $m \geq n$, a twice continuously differentiable function. At each iteration k, given the current iterate x_k, the objective function f is approximated by the norm of an affine model of F, resulting in a quadratic Taylor model for f with approximate Hessian matrix

$$m_k(x_k, s) = \frac{1}{2}\|F(x_k)\|^2 + \left(F'(x_k)^\top F(x_k)\right)^\top s + \frac{1}{2} s^\top H(x_k) s, \tag{2.7}$$

where F' is the Jacobian matrix of F, and $H(x_k) = F'(x_k)^\top F'(x_k)$ approximates the Hessian matrix $\nabla^2 f(x_k)$. We denote $m_k(x_k) := m_k(x_k, 0)$.

To address the invertibility issue discussed above, we consider instead the regularized model

$$m_{k,R}(x_k, s) := m_k(x_k, s) + \frac{\lambda_k}{2}\|s\|^2, \tag{2.8}$$

for $\lambda_k > 0$ a positive value, referred to as the regularization parameter. We write $m_{k,R}(x_k) := m_{k,R}(x_k, 0)$ for the value of the model at the zero step; this regularized model is minimized, possibly only approximately, to obtain a step s_k, which defines the candidate iterate $x_k + s_k$. Whether this step is actually accepted, however, is not automatic: at each iteration, we need to assess how well the model $m_{k,R}$ predicted the true behavior of the objective function f.

To this end, we compare the *actual* decrease of f with the decrease *predicted* by the model, through the ratio

$$\rho_k = \frac{\text{ared}}{\text{pred}} = \frac{f(x_k) - f(x_k + s_k)}{m_{k,R}(x_k) - m_{k,R}(x_k, s_k)}. \tag{2.9}$$

Intuitively, if the quadratic model is a good local approximation of f, then ρ_k should be close to one, since the model correctly anticipates the actual decrease. We therefore accept the step, setting $x_{k+1} = x_k + s_k$, whenever ρ_k exceeds a prescribed threshold $\eta_1 \in (0, 1)$; in this case, the iteration is called *successful*. Otherwise, the step is rejected, the iteration is called *unsuccessful*, and $x_{k+1} = x_k$. We now summarize the Levenberg–Marquardt method in Algorithm 2.

Algorithm 2 LM (Levenberg–Marquardt method)

Require: $x_0 \in \mathbb{R}^n$, $\lambda_0 > \lambda_{\min}$, $\lambda_{\min} > 0$, $0 < \eta_1 \le \eta_2 < 1$, $0 < \gamma_2 \le \gamma_1 < 1 < \gamma_3$, tolerance $\varepsilon > 0$, $k_{\max} \in \mathbb{N}$.

$k = 0$
while stopping criterion not satisfied **and** $k < k_{\max}$ **do**
  • **Initialization:** Define the model $m_{k,R}$ as in (2.8).
  • **Model minimization:** Compute a step s_k satisfying a sufficient decrease condition for the model.
  if $\|s_k\| \le \varepsilon$ **then**
    return x_k ▷ approximate stationary point
  end if
  • **Acceptance of the trial point and regularization parameter update:**
    Compute $\rho_k = \dfrac{f(x_k) - f(x_k + s_k)}{m_{k,R}(x_k, 0) - m_{k,R}(x_k, s_k)}$.
  if $\rho_k \ge \eta_1$ **then**
    $$x_{k+1} = x_k + s_k, \quad \lambda_{k+1} = \begin{cases} \max\{\lambda_{\min}, \gamma_2 \lambda_k\}, & \text{if } \rho_k \ge \eta_2, \\ \max\{\lambda_{\min}, \gamma_1 \lambda_k\}, & \text{if } \rho_k < \eta_2. \end{cases}$$
  else
    $$x_{k+1} = x_k, \quad \lambda_{k+1} = \gamma_3 \lambda_k.$$
  end if
  $k = k + 1$
end while

Since the model $m_{k,R}$ is λ_k-strongly convex, its decrease along a step computed as above is positive whenever $s_k \neq 0$, so that ρ_k is well defined. A step $s_k = 0$ corresponds to a stationary point of f; accordingly, the algorithm stops when $\|s_k\| \le \varepsilon$.

With this regularized model in place, our aim is now to integrate the Levenberg–Marquardt approach within the Proximal Gauss–Newton framework.

2.3.1 The Levenberg–Marquardt approach for the PGN method

The introduction of the Levenberg–Marquardt regularization for PGN serves two main purposes. On the one hand, it extends the Levenberg–Marquardt method, which has mainly been designed for smooth least-squares problems, to composite problems of the form

$$\min_{x\in\mathbb{R}^n} \Phi(x) := \frac{1}{2}\|F(x)\|^2 + J(x), \tag{2.10}$$

where J is a possibly non-smooth term. On the other hand, it makes PGN applicable in situations where the Gauss–Newton matrix $H(x_k) = F'(x_k)^\top F'(x_k)$ is singular or ill-conditioned. As discussed in Section 2.2, the Proximal Gauss–Newton method requires $H(x_k)$ to be invertible or, equivalently, $F'(x_k)$ to have full column rank. In practice this assumption may fail, or $H(x_k)$ may be nearly singular, so that each PGN step becomes ill-posed or numerically unstable. The idea is to replace $H(x_k)$ with the Levenberg–Marquardt metric $H_k = H(x_k)+\lambda_k I$, which is invertible for every $\lambda_k > 0$, without any assumption on $F'(x_k)$. This removes the difficulty described above, while preserving the adaptive choice of the regularization parameter.

This leads to the following Levenberg–Marquardt variant of the Proximal Gauss–Newton method. Given $x_0 \in \Omega$ and a sequence of regularization parameters $\{\lambda_k\}$ with $\lambda_k > 0$, we define the sequence $\{x_k\}$ by

$$x_{k+1} = \operatorname{prox}_J^{H_k}\left(x_k - H_k^{-1}F'(x_k)^\top F(x_k)\right). \tag{2.11}$$

The resulting method is summarized in Algorithm 3, where Φ denotes the objective function in (2.10) and $m_{k,R}$ the regularized model in (2.8).

Algorithm 3 Proximal Levenberg–Marquardt (PGN-LM)

Require: $x_0 \in \operatorname{dom} J$, $\lambda_0 > 0$, $\lambda_{\min} > 0$, $0 < \eta_1 \le \eta_2 < 1$, $0 < \gamma_2 \le \gamma_1 < 1 < \gamma_3$, tolerance $\varepsilon > 0$, $k_{\max} \in \mathbb{N}$.

$k = 0$
while stopping criterion not satisfied **and** $k < k_{\max}$ **do**
  $H_k = F'(x_k)^\top F'(x_k) + \lambda_k I$
  $\hat{x}_k = \operatorname{prox}_J^{H_k}\big(x_k - H_k^{-1} F'(x_k)^\top F(x_k)\big), \quad s_k = \hat{x}_k - x_k$
  if $\|s_k\| \le \varepsilon$ **then**
    return x_k ▷ approximate critical point
  end if
  $\rho_k = \dfrac{\Phi(x_k) - \Phi(x_k + s_k)}{\Phi(x_k) - \big(m_{k,R}(x_k, s_k) + J(x_k + s_k)\big)}$
  if $\rho_k \ge \eta_1$ **then**
    $x_{k+1} = x_k + s_k$ ▷ successful step
  else
    $x_{k+1} = x_k$ ▷ rejected step
  end if
  if $\rho_k \ge \eta_2$ **then**
    $\lambda_{k+1} = \max\{\lambda_{\min}, \gamma_2 \lambda_k\}$
  else if $\rho_k \ge \eta_1$ **then**
    $\lambda_{k+1} = \max\{\lambda_{\min}, \gamma_1 \lambda_k\}$
  else
    $\lambda_{k+1} = \gamma_3 \lambda_k$
  end if
  $k = k + 1$
end while

Our aim is now to establish the convergence of $\{x_k\}$ for suitable choices of the initial point x_0 and of the regularization parameters $\{\lambda_k\}$.

2.3.2 Convergence analysis

In this section, we establish the convergence properties of the proposed Proximal Levenberg–Marquardt method. To this end, we first present some results and notation that will be used in the subsequent analysis.

Lemma 2.2. *Suppose that $x_* \in \Omega$ is a local minimizer of Φ. Then, for every k,*

$$x_* = \operatorname{prox}_J^{H_k}\big(x_* - H_k^{-1} F'(x_*)^\top F(x_*)\big).$$

Proof. Since x_* is a local minimizer of Φ, it satisfies the first-order stationarity condition

$$-F'(x_*)^\top F(x_*) \in \partial J(x_*).$$

Adding $H_k x_*$ to both sides yields

$$H_k x_* - F'(x_*)^\top F(x_*) \in H_k x_* + \partial J(x_*),$$

that is,

$$H_k\big(x_* - H_k^{-1} F'(x_*)^\top F(x_*)\big) \in (H_k + \partial J)(x_*).$$

By the characterization of the proximity operator with respect to the metric induced by H_k, this is equivalent to

$$x_* = \operatorname{prox}_J^{H_k}\left(x_* - H_k^{-1} F'(x_*)^\top F(x_*)\right).$$

□

Lemma 2.3. *Let H_k be a symmetric positive definite matrix. For every vector v, the following inequality holds:*

$$\|v\|^2 \leq \|H_k^{-1}\|\, \|v\|_{H_k}^2.$$

Proof. By definition, the squared norm of v induced by H_k is given by the quadratic form $\|v\|_{H_k}^2 = v^\top H_k v$. Since H_k is symmetric, this quadratic form is bounded from below by the smallest eigenvalue of H_k:

$$v^\top H_k v \geq \lambda_{\min}(H_k)\|v\|^2.$$

Since H_k is positive definite, the spectral norm of its inverse is its largest eigenvalue, which is the reciprocal of the smallest eigenvalue of H_k:

$$\|H_k^{-1}\| = \lambda_{\max}(H_k^{-1}) = \frac{1}{\lambda_{\min}(H_k)}.$$

Substituting $\lambda_{\min}(H_k) = 1/\|H_k^{-1}\|$ in the previous inequality, we obtain

$$\|v\|_{H_k}^2 \geq \frac{1}{\|H_k^{-1}\|}\,\|v\|^2,$$

and multiplying both sides by $\|H_k^{-1}\| > 0$ yields the desired inequality. □

We are now ready to state the main convergence result, which concerns the simplified version of Algorithm 3 described above, in which every step is accepted.

Theorem 2.4 (Local convergence of the Proximal Levenberg–Marquardt method)**.** *Let $\Omega \subseteq \mathbb{R}^n$ be open, let $F\colon \Omega \to \mathbb{R}^m$ be differentiable, and let $J\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be proper, convex and lower semicontinuous. Consider*

$$\Phi(x) = \frac{1}{2}\|F(x)\|^2 + J(x),$$

and let $x_* \in \mathrm{dom}(J)$ be a local minimizer of Φ. Suppose that $U \subseteq \Omega$ is open and star-shaped with respect to x_*, and that the Jacobian $F' \colon U \to \mathbb{R}^{m\times n}$ is Lipschitz continuous on U with constant $L > 0$.

Set $\theta := \|F(x_*)\|$ and $\delta := \|F'(x_*)\|$. Let $H(x) := F'(x)^\top F'(x)$ be the unregularized matrix, and suppose that there exists a constant $\sigma_{\min} > 0$ such that

$$\lambda_{\min}\big(H(x)\big) \geq \sigma_{\min} \qquad \text{for all } x \in U.$$

Assume further that the regularization parameters are uniformly bounded, i.e., $\underline{\lambda} \leq \lambda_k \leq \overline{\lambda}$ for all k.

Define, for $r \geq 0$,

$$q(r) := \frac{1}{\sigma_{\min} + \underline{\lambda}} \left(\frac{rL(\delta + Lr)}{2} \right) + \frac{\overline{\lambda} + L\theta}{\overline{\lambda} + \sigma_{\min}},$$

and set

$$\bar{r} := \sup\{r \geq 0 : q(r) < 1\}. \tag{2.12}$$

If

$$\frac{L\theta}{\sigma_{\min}} < 1, \tag{2.13}$$

then $\bar{r} > 0$. Fix $r \in \mathbb{R}$, with $0 < r < \bar{r}$, such that $B_r(x_*) \subseteq U$. Then, for any initial guess $x_0 \in B_r(x_*)$, the sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by the Proximal Levenberg–Marquardt scheme

$$x_{k+1} = \mathrm{prox}_J^{H_k}\Big(x_k - H_k^{-1} F'(x_k)^\top F(x_k)\Big), \tag{2.14}$$

with $H_k := F'(x_k)^\top F'(x_k) + \lambda_k I$, is well defined, i.e., $x_k \in B_r(x_*)$ for all $k \geq 0$, and it holds

$$\|x_{k+1} - x_*\| \leq q_0^{k+1} \|x_0 - x_*\|, \qquad \forall k \geq 0, \tag{2.15}$$

where $q_0 := q(\|x_0 - x_*\|) < 1$.

Proof. Thanks to Lemma 2.3 and to the non-expansiveness of the proximity operator (Proposition 1.13), we have

$$\|x_{k+1} - x_*\|_{H_k} \leq \Big\| \big(x_k - H_k^{-1} F'(x_k)^\top F(x_k)\big) - \big(x_* - H_k^{-1} F'(x_*)^\top F(x_*)\big) \Big\|_{H_k}.$$

Using the definition of the preconditioned norm, $\|v\|_{H_k} = \|H_k^{1/2} v\|$, we factor out $H_k^{-1/2}$ to obtain

$$\|x_{k+1} - x_*\|_{H_k} \leq \left\| H_k^{-1/2} \underbrace{\big[H_k(x_k - x_*) - \big(F'(x_k)^\top F(x_k) - F'(x_*)^\top F(x_*)\big) \big]}_{D_k} \right\|.$$

Substituting $H_k = F'(x_k)^\top F'(x_k) + \lambda_k I$, we obtain

$$D_k = F'(x_k)^\top F'(x_k)(x_k - x_*) + \lambda_k(x_k - x_*) - F'(x_k)^\top F(x_k) + F'(x_*)^\top F(x_*).$$

By adding and subtracting $F'(x_k)^\top F(x_*)$ and grouping terms, we derive

$$D_k = F'(x_k)^\top \underbrace{[F'(x_k)(x_k - x_*) - F(x_k) + F(x_*)]}_{R_k} + \lambda_k(x_k - x_*) + \left(F'(x_*)^\top - F'(x_k)^\top\right) F(x_*).$$

Therefore, recalling that $\|v\|^2 \le \|H_k^{-1}\| \, \|v\|_{H_k}^2$,

$$\|x_{k+1} - x_*\| \le \|H_k^{-1}\|^{1/2} \, \|H_k^{-1}\|^{1/2} \left\| F'(x_k)^\top R_k + \lambda_k(x_k - x_*) + \left(F'(x_*)^\top - F'(x_k)^\top\right) F(x_*) \right\|.$$

Then the Lipschitz continuity of F' yields

$$\|x_{k+1} - x_*\| \le \|H_k^{-1}\| \left(\|F'(x_k)\| \frac{L}{2} \|x_k - x_*\|^2 + \lambda_k \|x_k - x_*\| + L \|x_k - x_*\| \, \|F(x_*)\| \right).$$

By spectral calculus, $\|H_k^{-1}\| \le \frac{1}{\sigma_{\min} + \lambda_k} =: \beta_k$. We show by induction that $x_k \in B_r(x_*)$ for every k. For $k = 0$ this holds by assumption. Assume now that $x_k \in B_r(x_*)$. Since $B_r(x_*) \subseteq U$, we have $\lambda_{\min}(H(x_k)) \ge \sigma_{\min}$, hence

$$\|H_k^{-1}\| = \frac{1}{\lambda_{\min}(H(x_k)) + \lambda_k} \le \frac{1}{\sigma_{\min} + \lambda_k},$$

and, by the Lipschitz continuity of F',

$$\|F'(x_k)\| \le \|F'(x_*)\| + L\|x_k - x_*\| \le \delta + Lr.$$

Using these bounds, together with $\|R_k\| \le \frac{L}{2}\|x_k - x_*\|^2$ and $\|(F'(x_*) - F'(x_k))^\top F(x_*)\| \le L\theta \, \|x_k - x_*\|$, we obtain

$$\begin{aligned} \|x_{k+1} - x_*\| &\le \frac{1}{\sigma_{\min} + \lambda_k} \left(\frac{L(\delta + Lr)}{2} \|x_k - x_*\|^2 + (\lambda_k + L\theta) \|x_k - x_*\| \right) \\ &\le \frac{1}{\sigma_{\min} + \lambda_k} \left(\frac{rL(\delta + Lr)}{2} + \lambda_k + L\theta \right) \|x_k - x_*\| =: q_k(r) \, \|x_k - x_*\|. \end{aligned}$$

We now bound $q_k(r)$ uniformly in k. Since $\lambda_k \ge \underline{\lambda}$, the first term satisfies

$$\frac{1}{\sigma_{\min} + \lambda_k} \frac{rL(\delta + Lr)}{2} \le \frac{1}{\sigma_{\min} + \underline{\lambda}} \frac{rL(\delta + Lr)}{2}.$$

Moreover, the function $\lambda \mapsto \frac{\lambda + L\theta}{\lambda + \sigma_{\min}}$ has derivative $\frac{\sigma_{\min} - L\theta}{(\lambda + \sigma_{\min})^2}$, which is positive by (2.13); it is therefore increasing, and since $\lambda_k \le \overline{\lambda}$,

$$\frac{\lambda_k + L\theta}{\lambda_k + \sigma_{\min}} \le \frac{\overline{\lambda} + L\theta}{\overline{\lambda} + \sigma_{\min}}.$$

Hence $q_k(r) \leq q(r)$ for every k. Note that $q(0) = \frac{\bar{\lambda}+L\theta}{\bar{\lambda}+\sigma_{\min}} < 1$ by (2.13), and that q is continuous and increasing, so that $\bar{r} > 0$ and $q(r) < 1$ for $0 < r < \bar{r}$. Therefore

$$\|x_{k+1} - x_*\| \leq q(r)\,\|x_k - x_*\| < \|x_k - x_*\| < r,$$

that is, $x_{k+1} \in B_r(x_*)$, which concludes the induction.

Finally, since $\|x_k - x_*\| \leq \|x_0 - x_*\|$ for every k, the same argument can be repeated with r replaced by $\|x_0 - x_*\|$, which gives $\|x_{k+1} - x_*\| \leq q_0\,\|x_k - x_*\|$ with $q_0 = q(\|x_0 - x_*\|) < 1$, and hence, by induction, $\|x_{k+1} - x_*\| \leq q_0^{k+1}\,\|x_0 - x_*\|$.

By the continuity of q, there exists a sufficiently small $r > 0$ such that $q(r) < 1$. In particular, for $\|x_k - x_*\| \leq r$, we have

$$\|x_{k+1} - x_*\| \leq q_k(\|x_k - x_*\|)\,\|x_k - x_*\| \leq q(\|x_k - x_*\|)\,\|x_k - x_*\|.$$

From this point onward, this implies that

$$\|x_{k+1} - x_*\| \leq q(\|x_k - x_*\|)\,\|x_k - x_*\| < \|x_k - x_*\|,$$

and, since q is non-decreasing,

$$q(\|x_{k+1} - x_*\|) \leq q(\|x_k - x_*\|).$$

Setting $q_0 := q(\|x_0 - x_*\|) < 1$, it follows by induction that

$$q(\|x_k - x_*\|) \leq q_0, \qquad \forall k \geq 0,$$

and consequently,

$$\|x_{k+1} - x_*\| \leq q_0 \|x_k - x_*\| \leq \cdots \leq q_0^{k+1} \|x_0 - x_*\|.$$

□

Despite these appealing theoretical properties, and the numerical stabilization provided by the Levenberg–Marquardt regularization, second-order proximal methods (both standard PGN and its regularized LM variant) share a critical computational bottleneck. At each outer iteration, computing the proximal step with respect to the variable metric induced by the surrogate Hessian ($H(x_k)$ or H_k) requires solving a large-scale linear system. The damping term in PGN-LM restores strict positive definiteness and improves conditioning, but it does nothing to reduce the cost of solving the linear systems with H_k required at each iteration, nor that of computing the scaled proximal step. In applications such as high-resolution image processing or fine-mesh PDE-constrained optimization, the dimension of the problem grows rapidly, making the repeated solution of these large-scale subproblems computationally prohibitive and poorly scalable. To overcome this prohibitive cost, the core objective of this thesis is to embed these second-order proximal strategies into a multilevel optimization framework. By exploiting a hierarchy of coarse and fine representations, multilevel strategies delegate the low-frequency error correction to computationally inexpensive coarse levels, drastically mitigating the burden of large-scale linear solves while retaining the fast convergence behavior of the underlying proximal solvers.

Chapter 3

Multilevel framework

Large-scale optimization problems, especially those arising from the discretization of continuous or infinite-dimensional models, often come with a natural hierarchy of resolutions. This chapter introduces the *multilevel framework*, a class of methods that exploits this hierarchy to accelerate convergence by combining information from coarser, computationally cheaper representations of the problem with the original fine-level formulation.

The intuition of exploiting a hierarchy of different resolutions stems originally from the numerical resolution of PDEs, where multiresolution techniques gave rise to the celebrated multigrid method [5]. This paradigm was later generalized to the continuous nonlinear optimization landscape through the pioneering work of Nash [8], who introduced the MG/OPT framework and thereby laid the foundations of the multilevel paradigm, showing how coarse-grid objective evaluations can be effectively used to compute cheap search directions for high-dimensional problems.

Most of the multilevel algorithms developed since then build directly on this seminal work, and are typically applied to smooth objective functions minimized by first-order methods, in the context of PDE-constrained problems where the objective f is assumed to be differentiable [6]. This framework has since been employed in a variety of applications, such as photoacoustic tomography [9] and phase retrieval [10].

More recently, this framework has also been extended to settings in which f is not differentiable. In particular, Parpas [11] proposed a multilevel Proximal Gradient algorithm for a class of composite optimization problems, combining the multilevel paradigm with proximal splitting techniques to handle non-smooth terms in the objective. Along the same lines, the work [13] developed multilevel proximal methods specifically tailored to image restoration problems.

This idea is particularly well suited to a broad class of optimization problems that arise from the discretization of problems posed in infinite-dimensional function spaces. In such settings, achieving a reliable discretization often requires working at a very large scale, which in turn leads to severe computational

bottlenecks. Multilevel algorithms overcome this issue precisely by exploiting the underlying continuous nature of the problem, decomposing the original high-dimensional task into a hierarchy of coarser, and hence computationally cheaper, subproblems.

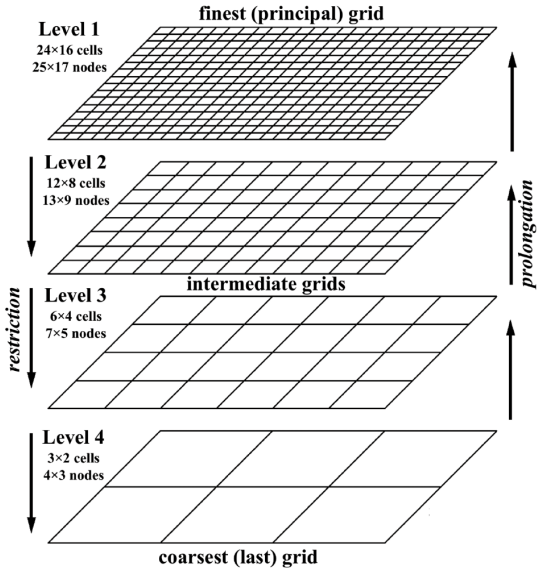


Figure 3.1: Visual scheme of the multilevel paradigm, illustrating the exchange of information across different resolution scales. Taken from [14].

3.1 Multilevel optimization in the smooth setting

Multilevel (ML) methods are iterative algorithms that generalize the multigrid philosophy to nonlinear continuous optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) := G(x) + J(x), \tag{3.1}$$

where $G, J \colon \mathbb{R}^n \to \mathbb{R}$ are both differentiable, so that f is differentiable as well. As the problem dimension grows, evaluating the optimization steps on the original high-fidelity model becomes computationally demanding. To mitigate these costs, ML frameworks exploit the inherent structure of the problem to build a hierarchy of reduced-dimension spaces. By minimizing these coarser representations of the objective function, the algorithm computes efficient, low-resolution update steps that accelerate the convergence on the finest level. To formally introduce the multilevel framework, we assume that the original optimization problem can be represented across a hierarchy of levels, indexed by $\ell = 1, \ldots, \ell_{\max}$. Specifically, we define the following components:

- **Dimensions:** a sequence of strictly increasing problem dimensions $n_1 < n_2 < \cdots < n_{\ell_{\max}} = n$. Here, $\ell_{\max}$ represents the finest level (the original problem), while $\ell = 1$ is the coarsest level.
- **Objective functions:** a collection of continuously differentiable functions $\{f^\ell\}_{\ell=1}^{\ell_{\max}}$, where each function operates on the corresponding reduced space, i.e., $f^\ell \colon \mathbb{R}^{n_\ell} \to \mathbb{R}$. At the finest level, the objective coincides with the original one: $f^{\ell_{\max}}(x) = f(x)$ for all $x \in \mathbb{R}^n$. Naturally, evaluating and minimizing $f^{\ell-1}$ is computationally cheaper than f^ℓ, due to the reduced dimensionality ($n_{\ell-1} < n_\ell$).
- **Transfer operators:** to allow communication between different scales of the hierarchy, we assume that a set of linear operators is available to transfer vectors between consecutive levels. Specifically, we require a *restriction* operator, mapping fine-level variables to the coarse space, and a *prolongation* operator, mapping coarse-level corrections back to the fine space.

We now define the mathematical framework used to formulate the coarse-level problem.

3.2 Coarse-level construction

Defining a coarse-grid problem requires specifying how information is exchanged between consecutive discretization levels. In this section, the presentation is restricted to the two-level case, and we introduce the notation that will be used throughout the rest of the manuscript. We denote by h the fine level and by H the coarse level, with N_h and $N_H < N_h$ degrees of freedom, respectively. Unless otherwise stated, the fine-level objective function, previously denoted f, will from now on be written as f^h, in order to make explicit its dependence on the fine-level discretization, while the corresponding quantities on the coarser grid are denoted with the superscript H. To transfer information between levels without losing mathematical coherence, we define the following two inter-grid operators.

Definition 3.1 (Coherent Information Transfer)**.** *The two operators $R \colon \mathbb{R}^{N_h} \to \mathbb{R}^{N_H}$ and $P \colon \mathbb{R}^{N_H} \to \mathbb{R}^{N_h}$ are called* Coherent Information Transfer *(CIT) operators if there exists a constant $\xi > 0$ such that*

$$P = \xi R^\top.$$

We denote by R the restriction operator, which transfers information from the fine level to the coarse level, and by P the prolongation operator, which performs the reverse transfer, from the coarse level back to the fine level. In other words, the prolongation operator is not chosen independently of the restriction operator, but is instead defined as its transpose, up to a positive scaling constant ξ. Now

that the transfer of information between levels has been established, the next step is to define the objective function at the coarse level. Our aim is to construct the coarse-level function in such a way that it helps to find the solution of the original, fine-level problem more rapidly. One way to guarantee this is to impose coherence between levels.

On the coarse grid, we consider the coarse counterparts $G^H, J^H \colon \mathbb{R}^{N_H} \to \mathbb{R}$ of G^h and J^h, obtained for instance by discretizing the same continuous problem on the coarser grid. Without first-order coherence, decreasing the coarse function $G^H + J^H$ does not necessarily lead to a decrease of $f^h = G^h + J^h$: the coarse level may produce a direction that is a descent direction for $G^H + J^H$ but not for the original fine-level function. This can happen because $G^H + J^H$ is, in general, only an approximation of f^h, and nothing guarantees *a priori* that its local behavior around the current iterate matches that of f^h.

To make the coarse level a reliable proxy for the fine-level problem, at least locally, it is natural to require that the two functions agree, to first order, at the point corresponding to the current fine-level iterate x_k^h, that is, that the gradient of the coarse function, evaluated at the restricted point $x_k^H := Rx_k^h$, coincides with the restriction of the fine-level gradient. In general, this identity does not hold for $G^H + J^H$, since the coarse function is constructed independently, and there is a mismatch between the two gradients. We measure this mismatch by defining the correction term

$$v_k^H := R\,\nabla f^h(x_k^h) - \nabla\big(G^H + J^H\big)(x_k^H). \tag{3.2}$$

Definition 3.2 (Coarse model for smooth functions)**.** *Let $G^H, J^H \colon \mathbb{R}^{N_H} \to \mathbb{R}$ be continuously differentiable approximations of G^h and J^h on the coarse grid. Given $x^h \in \mathbb{R}^{N_h}$, the coarse model f^H is defined as*

$$f^H = G^H + J^H + \langle v^H, \cdot \rangle, \tag{3.3}$$

where

$$v^H = R\,\nabla f^h(x^h) - \nabla\big(G^H + J^H\big)(Rx^h). \tag{3.4}$$

Adding the linear term $\langle v^H, \cdot \rangle$ to the coarse function $G^H + J^H$ makes it possible to enforce the so-called *first-order coherence* between the two levels.

Definition 3.3 (First-order coherence)**.** *The fine-level objective function f^h and the coarse-level objective function f^H are said to satisfy* first-order coherence *at x^h if*

$$\nabla f^H\big(Rx^h\big) = R\,\nabla f^h(x^h).$$

To better illustrate why the term v^H needs to be added, Figure 3.2 shows a comparison between the two cases, with and without this correction term, highlighting the resulting difference in behavior between the coarse and fine models.

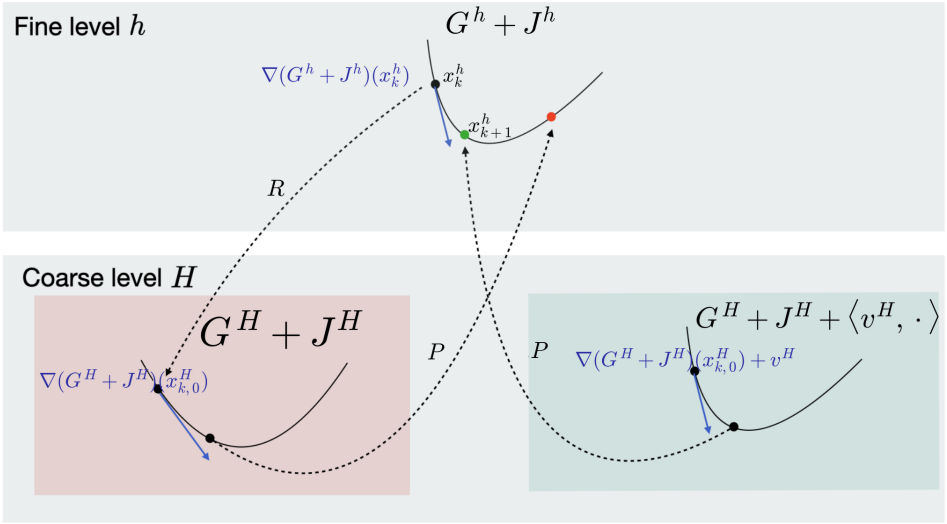


Figure 3.2: Effect of imposing first-order coherence between the fine-level function f^h and the coarse-level function f^H. *Left:* in the absence of coherence, a decrease of f^H does not guarantee a corresponding decrease of f^h. *Right:* once the correction term $v_k^H := R\nabla f^h(x_k^h) - \nabla(G^H + J^H)(x_{k,0}^H)$ is introduced, the graph of f^H is tilted around the point $x_{k,0}^H$, so that the two gradients coincide at that point and a decrease of f^H now translates into a decrease of f^h as well.

We now want to formalize the intuition that the coarse level effectively helps the fine one, by showing that the direction obtained at the coarse level, once transferred back to the fine level via P, is a descent direction for f^h at the current iterate x_k^h.

Lemma 3.4 (Descent direction for the fine-level function)**.** *Let R and P be CIT operators, with $P = \xi R^\top$ for some $\xi > 0$, and let f^H be the coarse model of Definition 3.2, built at the point x^h. Let $x_0^H = Rx^h$, and let x_m^H be the point obtained after m coarse iterations. Assume that the coarse correction $p_H := x_m^H - x_0^H$ is a descent direction for f^H at x_0^H, that is,*

$$\langle \nabla f^H(x_0^H), p_H \rangle < 0. \tag{3.5}$$

Then $P(p_H)$ is a descent direction for f^h at x^h.

Proof. Since $P = \xi R^\top$ and, by first-order coherence, $R\,\nabla f^h(x^h) = \nabla f^H(x_0^H)$,

$$\langle \nabla f^h(x^h), P(p_H) \rangle = \xi \, \langle R\,\nabla f^h(x^h), p_H \rangle = \xi \, \langle \nabla f^H(x_0^H), p_H \rangle < 0,$$

where the last inequality follows from (3.5) and $\xi > 0$. □

Remark 3.5 (On the descent assumption)**.** *Condition* (3.5) *is a mild requirement: it only asks that the coarse correction form an acute angle with the steepest*

descent direction $-\nabla f^H(x_0^H)$ *at the initial coarse point. Its validity depends on the convexity of the coarse model.*

Convex case. *If* f^H *is convex, condition* (3.5) *follows from a strict decrease of the coarse model, which is guaranteed by most coarse solvers, for instance gradient steps with a sufficiently small step size. Indeed, by convexity,*

$$\langle \nabla f^H(x_0^H), p_H \rangle \leq f^H(x_m^H) - f^H(x_0^H) < 0.$$

Non-convex case. *If* f^H *is not convex, a decrease of the coarse model no longer implies condition* (3.5)*: the coarse iterations may reach a point with a lower value of* f^H *along a path that starts in an ascent direction at* x_0^H*. Condition* (3.5) *can nevertheless be verified in practice at negligible cost: since* $\nabla f^H(x_0^H) = R\nabla f^h(x^h)$ *is available from the construction of the coarse model, checking* (3.5) *only requires one inner product, and a coarse correction that does not satisfy it can be discarded. Moreover, in our algorithm each coarse correction will be accepted only if it decreases the fine-level objective, through the line search of Algorithm 1; corrections that are not useful are therefore rejected, and cannot deteriorate the current iterate.*

We now describe how the coarse level is used in practice to correct the fine-level iterate. For simplicity, we suppose that the corrected coarse function is minimized by means of a standard gradient descent scheme, performing m_k steps with fixed step size α. The resulting coarse displacement is then prolongated back to the fine level and used to update the current iterate x^h, scaled by a step size α^H. The full procedure is summarized in Algorithm 4.

Algorithm 4 MultiLevel (ML) step for a smooth functional

if coarse correction is applied at the current iteration **then**
  $x_{k,0}^H = Rx^h$
  $v^H = R\nabla f^h(x^h) - \nabla(G^H + J^H)(x_{k,0}^H)$
  Choose step sizes $\alpha > 0$ and $\alpha^H > 0$
  Define the coarse gradient step operator $S(x) = x - \alpha\nabla f^H(x)$
  Apply m_k gradient steps: $x_{k,m_k}^H = \underbrace{(S \circ \cdots \circ S)}_{m_k \text{ times}}(x_{k,0}^H)$
  $\bar{x}^h = x^h + \alpha^H P(x_{k,m_k}^H - x_{k,0}^H)$
else
  $\bar{x}^h = x^h$
end if

The two-level scheme described above, which consists in going from the fine level down to the coarse level and then back to the fine level, is commonly referred to as a *V-cycle*. Such a V-cycle is illustrated in Figure 3.3.

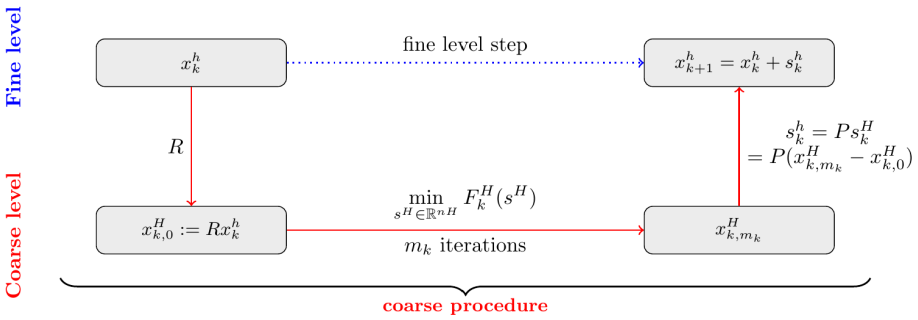


Figure 3.3: Scheme of the k-th iteration of the multilevel procedure. The top path (blue dotted line) shows the direct fine-level update, while the bottom path (red solid arrows) details the coarse-level correction: restriction to the coarse space (R), definition of the coarse function, m_k minimization steps, prolongation back to the fine space (P), and the final fine-level update. The elements highlighted in red require careful construction to guarantee a valid descent direction at the fine level.

So far, we have restricted our presentation to the two-level case. However, extending this construction to a full hierarchy of levels is not difficult, and follows quite naturally from what has already been introduced. Indeed, the coarse model built above is itself a smooth function, so that the very same procedure used to construct it from the fine level can be applied again, this time treating it as the fine level with respect to an even coarser one. In this way, one obtains a coarse model of the coarse model, first-order coherent with the latter in exactly the same sense as before.

From this perspective, the hierarchical levels are naturally organized in pairs: each coarse level is designed to accelerate the optimization process of the one immediately above it. The correction mechanism previously described (and illustrated in Figure 3.3) is therefore applied independently to every pair of consecutive levels. When multiple levels are available, the fundamental operation remains the same; the only variation lies in the sequence in which these pairs are visited.

Figure 3.4 shows a simplified scheme of one V-cycle iteration involving four levels, from the finest resolution J down to the coarsest one $J-3$. Starting from the fine level, the iterate is successively restricted to each coarser level (downward arrows) until the coarsest level is reached, where m minimization steps are performed. The resulting correction is then prolongated back to the level immediately above (upward arrows) and used to update the iterate; at that level, $m-1$ additional minimization steps are performed before the correction is prolongated further up, and so on until the fine level is reached again. In this way, each level except the coarsest one contributes exactly one iteration to the overall cycle, while the number of minimization steps decreases progressively as the cycle moves back up toward the fine level.

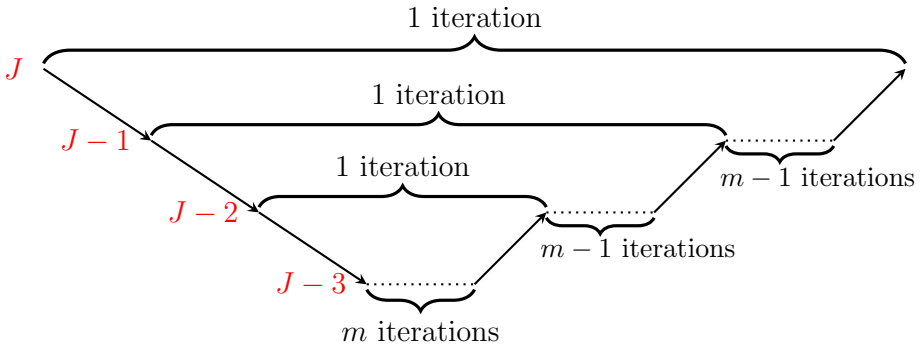


Figure 3.4: Simplified scheme of a V-cycle iteration. The downward arrows represent the restriction to coarser levels, while the upward arrows represent the prolongation and correction back to the finer levels.

One natural question is whether it is always convenient to descend to the coarse level. This is not necessarily the case, since the fine-level gradient $\nabla f^h(x_k^h)$ may lie partially or entirely in the null space of the restriction operator R. Consequently, $R\nabla f^h(x_k^h)$ may be small or even zero, although the fine-level gradient is not. In this situation, the coarse model may provide little to no useful correction.

For this reason, the coarse level is typically used when the restricted gradient is sufficiently large compared to the fine-level one. More precisely, the coarse model f^H is considered only when

$$\|R\nabla f^h(x_k^h)\| \geq \kappa_H \|\nabla f^h(x_k^h)\| \quad \text{and} \quad \|R\nabla f^h(x_k^h)\| > \varepsilon_H, \tag{3.6}$$

for some $\kappa_H \in (0, \min\{1, \|R\|\})$ and $\varepsilon_H \in (0, 1)$. The first condition ensures that the restricted gradient retains sufficient information from the fine level, while the second excludes the case in which the coarse gradient is already close to zero.

This criterion, however, requires computing the restricted gradient at every iteration, even when no coarse correction is performed. Moreover, since first-order coherence is only a local property, it may prevent consecutive coarse corrections that could otherwise be beneficial. A common alternative is therefore to prescribe a priori when the coarse level is used, following a predefined multilevel pattern, as in classical multigrid methods.

3.3 Non-smooth multilevel optimization

In the previous section, we described how the multilevel framework can be implemented in the case of a smooth objective function. In many applications of interest, however, the optimization problem is not smooth: this is the case, for instance, when a regularization term is added to the objective, as is typical in imaging and inverse problems, where such a term is often convex but non-differentiable (e.g., a total variation or an ℓ_1-type penalty). It is therefore natural to ask how

the multilevel strategy presented so far can be extended to handle this composite, non-smooth structure.

Typically, minimization problems of the following form are considered:

$$\min_x f(x) := G(x) + J(x). \tag{3.7}$$

This is precisely the situation we are interested in: recall from (2.1) that our objective function has the composite form $\Phi(x) = \frac{1}{2}\|F(x)\|^2 + J(x)$, where J is convex but, in general, non-smooth. Our aim is now to generalize the multilevel framework described above, originally designed for smooth objective functions, to the non-smooth setting. In the non-smooth setting, two types of coarse models can be defined: *smooth* and *non-smooth* ones. The smooth coarse model has the advantage that cheap first-order methods, such as gradient descent, can be used for its minimization. The non-smooth coarse model, on the other hand, may be preferable precisely because it more closely resembles the fine-level objective, which can lead to better-quality coarse steps. We start by investigating the smooth coarse model.

3.3.1 Smooth coarse model

The rich understanding that the optimization community has developed for smooth problems has motivated the design of *smoothing frameworks*, which allow non-smooth optimization problems to be approached, at least locally or approximately, by means of smooth surrogates, thereby making the full machinery of smooth optimization, including the multilevel strategy discussed so far, available also in the non-smooth case.

Among the various smoothing techniques available, one of the most natural and widely used is the *Moreau envelope*, which is closely related to the proximity operator introduced in Chapter 1.

Definition 3.6 (Moreau envelope)**.** *Let $g \in \Gamma_0(\mathbb{R}^n)$, and let $\gamma > 0$. The* Moreau envelope *of g with parameter γ is the function ${}^\gamma g\colon \mathbb{R}^n \to \mathbb{R}$ defined by*

$${}^\gamma g(x) := \min_{z\in\mathbb{R}^n} \left\{ g(z) + \frac{1}{2\gamma}\|x - z\|^2 \right\}.$$

The unique minimizer of the above problem is $\operatorname{prox}_{\gamma g}(x)$, so that

$${}^\gamma g(x) = g\big(\operatorname{prox}_{\gamma g}(x)\big) + \frac{1}{2\gamma}\left\|x - \operatorname{prox}_{\gamma g}(x)\right\|^2.$$

The Moreau envelope is always continuously differentiable, with $\frac{1}{\gamma}$-Lipschitz gradient, and it always lies below g, since taking $z = x$ in its definition gives ${}^\gamma g \le g$. However, it is not necessarily uniformly close to g from below. A relevant example is the indicator function of a nonempty closed convex set C, for which

${}^{\gamma}\iota_C = \frac{1}{2\gamma}\operatorname{dist}(\cdot, C)^2$ is finite everywhere, whereas $\iota_C = +\infty$ outside C, so that the gap between the two functions is unbounded.

A natural question, regardless of the smoothing technique employed, is therefore how accurately the resulting smooth surrogate approximates the original function. To address this question, we introduce the notion of *smoothable convex function*, which explicitly bounds the approximation error from both above and below.

Definition 3.7 (Smoothable convex function [16])**.** *Let $J \in \Gamma_0(\mathbb{R}^n)$, and let $X \subseteq \mathbb{R}^n$ be a closed convex set. The function J is said to be* (μ, η, K)-smoothable over X *if there exist $\eta_1, \eta_2 \geq 0$ with $\eta_1 + \eta_2 = \eta > 0$ such that, for every $\gamma > 0$, there exists a continuously differentiable convex function $J_\gamma \colon \mathbb{R}^n \to \mathbb{R}$ satisfying:*

(i) for every $x \in X$, $\quad J(x) - \eta_1\gamma \leq J_\gamma(x) \leq J(x) + \eta_2\gamma$;

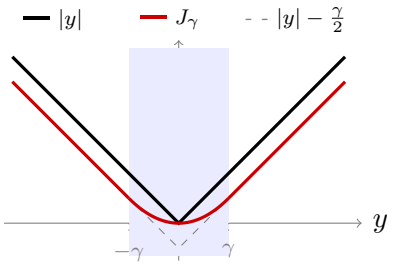
(ii) J_γ has Lipschitz continuous gradient over X, with Lipschitz constant at most $K + \mu/\gamma$.

In this case, J_γ is called a γ-smooth approximation *of J over X, with parameters (μ, η, K).*

The role of γ in this definition deserves some comment. Condition (i) bounds the approximation error $J_\gamma(x) - J(x)$ between $-\eta_1\gamma$ and $\eta_2\gamma$, so that J_γ never deviates from J by more than $\eta\gamma$. As $\gamma \to 0^+$, this error vanishes and J_γ converges to J. At the same time, condition (ii) bounds the Lipschitz constant of ∇J_γ by $K + \mu/\gamma$, which grows as γ decreases. The two conditions therefore capture a trade-off inherent to any smoothing scheme: small values of γ give an accurate but potentially ill-conditioned approximation, while larger values of γ sacrifice accuracy in exchange for a smoother, better-conditioned surrogate. The parameter γ thus controls, simultaneously, how close J_γ is to J and how regular it is.

Example 3.8 (Huber approximation of the absolute value)**.** *To illustrate this trade-off, consider the absolute value function $J(y) = |y|$, which is not differentiable at the origin. A standard γ-smooth approximation is given by the* Huber function*:*

$$J_\gamma(y) = \begin{cases} \dfrac{y^2}{2\gamma}, & |y| \leq \gamma, \\ |y| - \dfrac{\gamma}{2}, & |y| > \gamma. \end{cases}$$



This function is continuously differentiable everywhere and satisfies the bounds

$$J(y) - \frac{\gamma}{2} \leq J_\gamma(y) \leq J(y),$$

which fits our definition with parameters $\eta_1 = 1/2$ and $\eta_2 = 0$.

Having introduced the notion of smoothable convex function, we are now in a position to define the coarse model in the non-smooth setting. The idea is to combine the two ingredients discussed so far: on the one hand, the smoothing of the non-smooth term J^h, via its γ-smooth approximation $J^h_{\gamma_h}$; on the other, the first-order coherence correction v^H, already introduced in the smooth case, which ensures that a decrease of the coarse model translates into a decrease of the fine-level objective.

Definition 3.9 (Smooth coarse model for non-smooth functions)**.** *A continuously differentiable coarse model f^H is defined, for a given point $x^h \in \mathbb{R}^{N_h}$, as*

$$f^H = G^H + J^H_{\gamma_H} + \langle v^H, \cdot \rangle,$$

where

$$v^H = R\left(\nabla G^h(x^h) + \nabla J^h_{\gamma_h}(x^h)\right) - \left(\nabla G^H(Rx^h) + \nabla J^H_{\gamma_H}(Rx^h)\right).$$

By construction, f^H is smooth, since it is built from the smoothed regularization term $J^H_{\gamma_H}$ rather than from J^H itself. Moreover, exactly as discussed in Section 3.1 for the smooth case, f^H satisfies first-order coherence with the fine-level objective $f^h_{\gamma_h} = G^h + J^h_{\gamma_h}$ at the point x^h: by construction of the correction term v^H, the gradients of the coarse model and of the smoothed fine-level objective coincide, once restricted, at the current point. As a consequence, by Lemma 3.4, the prolongated coarse correction is a descent direction for the *smoothed* fine-level objective $f^h_{\gamma_h} = G^h + J^h_{\gamma_h}$.

This does not imply, however, that it is a descent direction for the original objective $f^h = G^h + J^h$, which is not differentiable: a direction along which the smoothed function decreases may still increase f^h, for instance near a point of non-differentiability of J^h. What can be guaranteed is an approximate decrease, with an error controlled by the smoothing parameter. Indeed, the following result shows that f^h does not increase by more than $(\eta_1+\eta_2)\gamma_h$, a quantity that vanishes as $\gamma_h \to 0$.

3.3.2 Fine-level decrease

We now specify what is required from the minimization scheme used at the coarse level. Let Φ^H denote an operator representing one step of some chosen optimization method applied to the coarse model f^H, for instance a gradient step $\Phi^H(x) = x - \alpha\nabla f^H(x)$, or a Proximal Gradient step, depending on whether f^H is smooth or not. Applying this operator repeatedly, starting from the restricted point $x^H_0 := Rx^h$, produces the coarse iterates

$$x^H_{\ell+1} = \Phi^H(x^H_\ell), \qquad \ell = 0, \dots, m-1.$$

For this construction to be useful within the multilevel scheme, we require the resulting sequence to yield a genuine decrease of the coarse objective after a finite number of steps. This is formalized in the following assumption.

Assumption 1 (Coarse model decrease)**.** *Let $(\Phi_\ell^H)_{\ell\in\mathbb{N}}$ be a sequence of operators as above. We assume that there exists an integer $m > 0$ such that, setting*

$$x_m^H := \Phi_{m-1}^H \circ \cdots \circ \Phi_0^H(x_0^H),$$

it holds $f^H(x_m^H) < f^H(x_0^H)$.

Having introduced this assumption, we are now in a position to prove that the fine-level objective indeed decreases, up to a small error, as stated in the following lemmas, taken from [12].

Lemma 3.10 (Descent direction for the fine-level smoothed function)**.** *Let R and P be CIT operators, with $P = \xi R^\top$ for some $\xi > 0$, and let f^H be the smooth coarse model of Definition 3.9, built at the point x^h. Let $x_0^H = Rx^h$, and let x_m^H be the point obtained after m coarse iterations. Assume that the coarse correction $p_H := x_m^H - x_0^H$ is a descent direction for f^H at x_0^H, that is,*

$$\langle \nabla f^H(x_0^H), p_H\rangle < 0. \tag{3.8}$$

Then $P(p_H)$ is a descent direction for $G^h + J_{\gamma_h}^h$ at x^h.

Proof. The proof follows analogously to Lemma 3.4 by replacing the smooth function with the smoothed objective $G^h + J_{\gamma_h}^h$ and its corresponding coarse model f^H. □

We now choose a step size $\bar\alpha > 0$ that prevents too large corrections from the coarse level, by requiring that

$$\big(G^h + J_{\gamma_h}^h\big)\big(x^h + \bar\alpha\, P(x_m^H - x_0^H)\big) \le \big(G^h + J_{\gamma_h}^h\big)(x^h). \tag{3.9}$$

Such a step size exists, since $P(x_m^H - x_0^H)$ is a descent direction for $G^h + J_{\gamma_h}^h$ by Lemma 3.10.

Lemma 3.11 (Approximate fine-level decrease)**.** *If the assumptions of Lemma 3.10 hold,*

$$f^h\big(x^h + \bar\alpha\, P\left(x_m^H - x_0^H\right)\big) \le f^h(x^h) + (\eta_1 + \eta_2)\gamma_h. \tag{3.10}$$

Proof. This follows directly from the definition of a smoothable convex function:

$$\begin{aligned} f^h(x^h + \bar\alpha\, P(x_m^H - x_0^H)) &\le f_{\gamma_h}^h(x^h + \bar\alpha\, P(x_m^H - x_0^H)) + \eta_1\gamma_h \\ &\le f_{\gamma_h}^h(x^h) + \eta_1\gamma_h \\ &\le f^h(x^h) + (\eta_1 + \eta_2)\gamma_h. \end{aligned}$$

□

This result shows that a coarse-level minimization step leads to a decrease of f^h, up to the constant $(\eta_1+\eta_2)\gamma_h$, which can be made arbitrarily small by driving γ_h to zero.

3.3.3 Non-smooth coarse model

So far, we have considered a *smooth* coarse model, obtained by replacing the non-smooth term J^H with its γ-smooth approximation $J^H_{\gamma_H}$. As anticipated, however, this is not the only viable choice: it is also possible to work with a *non-smooth coarse model*. To guarantee that the fine-level objective decreases, an additional assumption on the smoothing is required. We now introduce the class of smoothing schemes for which this guarantee holds.

Assumption 2. ***Smoothing type.*** *Let $\gamma_H > 0$. Let J^H be a smoothable convex function with γ_H-smooth approximation $J^H_{\gamma_H}$. We assume that there exists $\eta > 0$ such that the following holds:*

$$(\forall x \in X) \quad J^H(x) - \eta\gamma_H \leq J^H_{\gamma_H}(x) \leq J^H(x). \tag{3.11}$$

Having introduced Assumption 2, we are now in a position to define the non-smooth coarse model, for the case in which the fine-level regularization term is treated without smoothing.

Definition 3.12 (Non-smooth coarse model)**.** *The non-smooth coarse model f^H is defined, for a given point $x^h \in \mathbb{R}^{N_h}$, as*

$$f^H = G^H + J^H + \langle v^H, \cdot \rangle, \tag{3.12}$$

where

$$v^H = R\left(\nabla G^h(x^h) + \nabla J^h_{\gamma_h}(x^h)\right) - \left(\nabla G^H(Rx^h) + \nabla J^H_{\gamma_H}(Rx^h)\right). \tag{3.13}$$

When the non-smooth coarse model is used, first-order coherence (Definition 3.3) is imposed between the smoothed version of the fine-level objective and the coarse-level objective, rather than between the fine- and coarse-level objectives directly. This choice may appear counterintuitive at first, since the coarse model itself is non-smooth; however, first-order coherence is a differential condition, and can therefore only be formulated between smooth quantities. Consequently, even when working with a non-smooth coarse model, the correction term v^H is still computed using the gradient of the smoothed fine-level objective, $G^h + J^h_{\gamma_h}$, rather than the (in general non-existent) gradient of $G^h + J^h$. Note, in particular, that the coarse-level term $J^H_{\gamma_H}$ appearing in v^H is itself smoothed; this is necessary since the correction term requires a gradient of the coarse regularization as well, while the coarse model f^H in (3.12) retains the true, non-smooth term J^H.

3.3.4 Fine-level decrease for non-smooth coarse models

We now describe how the correction computed at the coarse level, using the non-smooth coarse model of Definition 3.12, translates into a descent direction at the fine level, following [12].

Lemma 3.13 (Descent direction for the fine-level smoothed function)**.** *Let R and P be CIT operators, with $P = \xi R^\top$ for some $\xi > 0$, and let f^H be the non-smooth coarse model of Definition 3.12, built at the point x^h, with smoothed version*

$$f^H_{\gamma_H} := G^H + J^H_{\gamma_H} + \langle v^H, \cdot \rangle.$$

Let $x^H_0 = Rx^h$, and let x^H_m be the point obtained after m coarse iterations. Assume that the coarse correction $p_H := x^H_m - x^H_0$ is a descent direction for $f^H_{\gamma_H}$ at x^H_0, that is,

$$\langle \nabla f^H_{\gamma_H}(x^H_0), p_H \rangle < 0. \tag{3.14}$$

Then $P(p_H)$ is a descent direction for $G^h + J^h_{\gamma_h}$ at x^h, that is,

$$\langle \nabla \big(G^h + J^h_{\gamma_h}\big)(x^h), P(p_H) \rangle < 0.$$

Proof. Let $f^h_{\gamma_h} := G^h + J^h_{\gamma_h}$. By the first-order coherence between the smoothed functions,

$$\nabla f^H_{\gamma_H}(x^H_0) = R\, \nabla f^h_{\gamma_h}(x^h).$$

Since $P = \xi R^\top$, we have $P^\top = \xi R$. Therefore,

$$\begin{aligned} \langle \nabla f^h_{\gamma_h}(x^h), P(p_H) \rangle &= \langle P^\top \nabla f^h_{\gamma_h}(x^h), p_H \rangle = \xi \, \langle R\, \nabla f^h_{\gamma_h}(x^h), p_H \rangle \\ &= \xi \, \langle \nabla f^H_{\gamma_H}(x^H_0), p_H \rangle < 0, \end{aligned}$$

where the last inequality follows from (3.14) and $\xi > 0$. □

Remark 3.14. *In our setting the coarse model is not convex in general: a decrease of the function after m iterations does not imply that the initial direction p_H is a descent direction. For this reason, we assume the descent condition* (3.14) *directly.*

Since $G^h + J^h_{\gamma_h}$ is continuously differentiable and, by Lemma 3.13, $P(x^H_m - x^H_0)$ is a descent direction for it at x^h, there exists a step size $\bar{\alpha} > 0$ such that

$$\big(G^h + J^h_{\gamma_h}\big)\big(x^h + \bar{\alpha} P(x^H_m - x^H_0)\big) < \big(G^h + J^h_{\gamma_h}\big)(x^h),$$

and we recover the fine-level decrease property.

Lemma 3.15 (Approximate fine-level decrease)**.** *If the assumptions of Lemma 3.13 hold,*

$$f^h\left(x^h + \bar{\alpha}\, P\left(x^H_m - x^H_0\right)\right) \leq f^h(x^h) + (\eta_1 + \eta_2)\gamma_h. \tag{3.15}$$

Proof. The proof follows exactly the same argument as the proof of Lemma 3.11, since by Lemma 3.13, $P(x^H_m - x^H_0)$ is a descent direction for $f^h_{\gamma_h} := G^h + J^h_{\gamma_h}$ at x^h. □

This result allows us to use a non-smooth coarse model at the coarse level, which is expected to provide a more faithful approximation of the fine-level objective function than a smoothed one. This is particularly worth exploiting whenever an explicit expression for the proximity operator of J^H is available.

In the non-smooth setting, the multilevel step follows the structure of Algorithm 4, with two modifications. First, the first-order coherence term v^H is computed with the smoothed functions, as in (3.13). Second, if the coarse model retains the non-smooth term J^H, the operator S must be replaced by one iteration of a method suited to non-smooth composite problems, such as the Proximal Gauss–Newton method.

The decision of when to descend to the coarse level also requires some care, since criterion (3.6) relies on the gradient of the fine-level objective, which is no longer defined. Two strategies are possible. The first one is to avoid any adaptive criterion and use a fixed multilevel pattern, as in classical multigrid. The second one is to apply criterion (3.6) to the smoothed approximation $f^h_{\gamma_h} = G^h + J^h_{\gamma_h}$ of the objective function, that is, to the same function used to enforce first-order coherence.

Having introduced the multilevel framework, our aim is now to combine it with the Proximal Gauss–Newton method, in order to accelerate the resolution of the composite optimization problem (P) introduced in the previous chapters.

3.4 Multilevel Proximal Gauss–Newton method

Our contribution is to design a second-order method whose computational cost does not grow too severely with the problem dimension N_h. To this end, in this section we combine the Proximal Gauss–Newton method (Section 2.2) with a multilevel framework, obtaining a method which we refer to as ML-PGN. We consider the optimization problem

$$\min_{x\in\mathbb{R}^n} \ \frac{1}{2}\|F(x)\|^2 + J(x).$$

As seen in the previous sections, multilevel methods are recursive, so it suffices to describe the setting with two levels only. We consider the problem obtained from the discretization of the continuous problem on a fine grid, with N_h degrees of freedom. The resulting finite-dimensional problem,

$$\min_{x^h\in\mathbb{R}^{N_h}} \ f^h(x^h) := G^h(x^h) + J^h(x^h), \qquad G^h(x^h) := \frac{1}{2}\|F^h(x^h)\|^2, \tag{3.16}$$

is regarded as the problem on the fine level, where F^h and J^h denote the discretized residual and penalty term, respectively. On a coarser grid, with $N_H < N_h$ degrees of freedom, we consider the corresponding coarse quantities f^H, J^H and

$G^H(x^H) := \frac{1}{2}\|F^H(x^H)\|^2$. Given the current fine iterate x^h, the coarse model is

$$\min_{x^H \in \mathbb{R}^{N_H}} \; f^H(x^H) := G^H(x^H) + J^H(x^H) + \langle v^H, x^H \rangle, \tag{3.17}$$

which is cheaper to solve and is used to compute corrections for the fine-level iterates. Here, v^H is the first-order correction term of Definition 3.12.

The two levels play different roles. On the fine level, where the original problem is posed, we apply the Proximal Gauss–Newton method. On the coarse level, instead, we are free to choose the method used to minimize the coarse model: for instance, if the coarse model is smooth, a simple gradient descent method can be used, whereas a Gauss–Newton method allows us to exploit second-order information also on the coarse level. We now present a sketch of the multilevel algorithm associated with PGN, ML-PGN, in the two-level setting, with fine level h and coarse level H. Instead of using an adaptive criterion to decide when to descend to the coarse level, we adopt a fixed multilevel pattern: a coarse correction is performed at each of a prescribed number $K_{\max}$ of V-cycles, after which the method proceeds with fine-level iterations only.

Algorithm 5 Two-level multilevel Proximal Gauss–Newton method

Require: Initial iterate $x_0^h \in C$, tolerance $\varepsilon > 0$, coarse iterations m_c, cycles $K_{\max}$, terminal steps m_{term}.
Ensure: Feasible approximate solution $x_*^h \in C$.

for $k = 0, \dots, K_{\max} - 1$ **do** ▷ Fine-level presmoothing
  $\tilde{x}_k^h \leftarrow \text{PGN}(x_k^h; 1)$ subject to C
  if $\|\tilde{x}_k^h - x_k^h\|_2 < \varepsilon$ **then**
    return $\tilde{x}_k^h$
  end if ▷ Coarse model construction
  $x_0^H \leftarrow R\,\tilde{x}_k^h$
  $v^H \leftarrow R\nabla f^h(\tilde{x}_k^h) - \nabla f^H(x_0^H)$
  Define $\hat{f}^H(x) \coloneqq f^H(x) + \langle v^H, x \rangle$ ▷ Coarse correction step
  Starting from x_0^H, perform m_c iterations on $\hat{f}^H$, obtaining $x_{m_c}^H$
  $s_H^\star \leftarrow x_{m_c}^H - x_0^H$ ▷ coarse correction
  $d_k^h \leftarrow P\,s_H^\star$ ▷ prolongated correction
  Determine the step length α_k via line search
  $x_{k+1}^h \leftarrow \tilde{x}_k^h + \alpha_k d_k^h$
end for ▷ Terminal fine-level convergence
$x_*^h \leftarrow \text{PGN}(x_{K_{\max}}^h; m_{\text{term}})$ subject to C
return x_*^h

The step length α_k along the prolongated coarse correction $d_k^h = P\,s_H^\star$ is chosen by a backtracking line search analogous to Algorithm 1, in which the Armijo condition is replaced by a simple decrease condition, since f^h is not differentiable:

starting from $\alpha_k = 1$, the step is halved until $f^h(\tilde{x}_k^h + \alpha_k d_k^h) < f^h(\tilde{x}_k^h)$ is satisfied. If no such step is found within a maximum of 15 reductions, the coarse correction is discarded and $x_{k+1}^h = \tilde{x}_k^h$.

In the same way, we can define a multilevel variant of PGN-LM (Section 2.3.1), which we refer to as ML-LM: on the fine level, the Proximal Gauss–Newton method is replaced by PGN-LM, so that the approximate Hessian $(F^h)'(x^h)^\top (F^h)'(x^h)$ is replaced by its regularized version $(F^h)'(x^h)^\top (F^h)'(x^h) + \lambda_k I$, while the coarse level is treated as before.

Chapter 4

Numerical implementation and results

In this chapter, we assess numerically the methods introduced in the previous chapters: the Proximal Gauss–Newton method, its Levenberg–Marquardt regularization, and their multilevel variants. In all the experiments, we consider problem (2.1) with $J = \iota_C$, where C is a box, that is, the box-constrained nonlinear least-squares problem

$$\min_{\mathbf{x} \in C} \ \frac{1}{2}\|F(\mathbf{x})\|^2.$$

As a specific instance, we choose F as the residual of the discretization of a nonlinear elliptic boundary value problem, which also provides a natural hierarchy of discretizations on which the multilevel strategy can be applied. We first describe the problem and its discretization, and then present the numerical results, starting from the one-dimensional case and then moving to the two-dimensional one. In particular, for both the one- and the two-dimensional case, we consider two different settings: one in which the solution lies in the interior of the constraint box, and one in which the constraint is active. In the first setting, the projection can be avoided at the coarse level, so we use a smooth coarse model, setting $J^H = 0$. In the second setting, instead, the constraint must be taken into account also at the coarse level, and we set $J^H = \iota_{C_H}$, the indicator function of a suitable coarse constraint set C_H, which results in a non-smooth coarse model.

4.1 Smooth coarse problem

In this section, we describe how the test problems are constructed and how the methods are configured in the numerical experiments, considering the setting in which the box constraint is not active at the solution. In this case, we neglect the constraint at the coarse level and set $J^H = 0$, so that the coarse model is smooth.

4.1.1 1D case

We start from the one-dimensional case: let $\Omega = (0, 1)$ and consider the boundary value problem

$$\begin{cases} -u''(y) + e^{u(y)} = f(y) & \text{in } \Omega, \\ u(0) = u(1) = 0. \end{cases} \tag{4.1}$$

Using standard second-order finite differences on a uniform grid with step size $h = \frac{1}{n+1}$, the discretized problem at the internal nodes $y_i = ih$, for $i = 1, \dots, n$, becomes

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + e^{u_i} = f_i, \tag{4.2}$$

with $u_0 = u_{n+1} = 0$ by the homogeneous boundary conditions.

Given a set of grid points $\{y_i\}_{i=1}^n$ in Ω, the discrete counterpart of Problem (4.1) is then formulated as the following optimization task:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \ \|A\mathbf{x} + e^{\mathbf{x}} - \mathbf{z}\|^2 + \iota_C(\mathbf{x}), \tag{4.3}$$

where $x_i \approx u(y_i)$ for $i = 1, \dots, n$; $A \in \mathbb{R}^{n \times n}$ is the discrete Laplacian matrix, with entries $A_{ii} = 2/h^2$, $A_{i,i\pm1} = -1/h^2$, and $A_{ij} = 0$ otherwise, and $e^{\mathbf{x}}$ is understood component-wise; $\mathbf{z}$ collects the values $f(y_i)$; and ι_C is the indicator function of the box constraint set C.

The introduction of the constraint set C allows us to incorporate additional physical or a priori information on the solution. For instance, box constraints can enforce non-negativity or prescribe known lower and upper bounds on the state variable, as may arise in applications involving temperature, concentration, density, or potential fields. The problem can therefore be formulated as a constrained nonlinear least-squares problem, providing a natural setting for proximal optimization methods. We identify the nonlinear residual mapping as

$$F(\mathbf{x}) := \sqrt{2}\left(A\mathbf{x} + e^{\mathbf{x}} - \mathbf{z}\right). \tag{4.4}$$

We now check that the problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \ \frac{1}{2}\|F(\mathbf{x})\|^2 + \iota_C(\mathbf{x}), \qquad \text{with } C = [a, b]^n,$$

satisfies the assumptions of the local convergence results of PGN (Theorem 2.1) and of PGN-LM (Theorem 2.4).

- *Assumptions on J.* Since C is nonempty, closed and convex, $J = \iota_C$ is proper, lower semicontinuous and convex.
- *Existence of a minimizer.* Since C is compact and $\frac{1}{2}\|F\|^2$ is continuous, the problem admits a global minimizer $\mathbf{x}_* \in C = \operatorname{dom} J$.

- *Differentiability of F.* F is of class C^∞, with Jacobian

$$F'(\mathbf{x}) = \sqrt{2}\,\big(A + D(\mathbf{x})\big), \qquad D(\mathbf{x}) := \operatorname{diag}\big(e^{x_1}, \dots, e^{x_n}\big).$$

- *Lipschitz continuity of F'.* Let U be a bounded open neighborhood of $\mathbf{x}_*$, convex (hence star-shaped with respect to $\mathbf{x}_*$), and let M be such that $|x_i| \le M$ for all $\mathbf{x} \in U$. Since $F'(\mathbf{x}) - F'(\mathbf{y}) = \sqrt{2}\,\operatorname{diag}\big(e^{x_i} - e^{y_i}\big)$, the mean value theorem gives, for $\mathbf{x}, \mathbf{y} \in U$,

$$\|F'(\mathbf{x}) - F'(\mathbf{y})\|_2 = \sqrt{2}\max_i |e^{x_i} - e^{y_i}| \le \sqrt{2}\,e^M \|\mathbf{x} - \mathbf{y}\|_\infty \le \sqrt{2}\,e^M \|\mathbf{x} - \mathbf{y}\|_2.$$

  Hence F' is Lipschitz continuous on U, with constant $L = \sqrt{2}\,e^M$. Note that F' is not globally Lipschitz continuous, due to the exponential term, but the local results only require this property on a neighborhood of $\mathbf{x}_*$.

- *Injectivity of F' and lower bound on H.* The discrete Laplacian A is symmetric positive definite, and $D(\mathbf{x})$ is diagonal with positive entries. Hence the square matrix $F'(\mathbf{x})$ is symmetric positive definite, and in particular invertible, for every $\mathbf{x}$; this gives the injectivity assumption of Theorem 2.1. Moreover, since $\lambda_{\min}(A + D(\mathbf{x})) \ge \lambda_{\min}(A) > 0$,

$$\lambda_{\min}\big(H(\mathbf{x})\big) = 2\,\lambda_{\min}\big(A+D(\mathbf{x})\big)^2 \ge 2\,\lambda_{\min}(A)^2 =: \sigma_{\min} > 0 \qquad \text{for all } \mathbf{x} \in \mathbb{R}^n,$$

  which is the uniform bound required in Theorem 2.4, here valid on the whole space.

- *Small residual condition.* Both theorems require the residual at the solution $\mathbf{x}_*$ to be small: $\big[(1+\sqrt{2})\kappa + 1\big]\theta\beta^2 L < 1$ for PGN and $L\theta < \sigma_{\min}$ for PGN-LM, with $\theta = \|F(\mathbf{x}_*)\|$. When the constraint is inactive, the data can be constructed as $\mathbf{z} = A\mathbf{u}^\star + e^{\mathbf{u}^\star}$, with $\mathbf{u}^\star \in C$ a prescribed vector, so that $F(\mathbf{u}^\star) = 0$. Since the objective is nonnegative and vanishes at $\mathbf{u}^\star$, the solution is then $\mathbf{x}_* = \mathbf{u}^\star$, so that $\theta = 0$ and both conditions hold. When the constraint is active, instead, the constrained minimizer does not, in general, solve the discrete system, so that $F(\mathbf{x}_*) \neq 0$, and the size of the residual is not known a priori. The small residual conditions can then not be guaranteed, and the local convergence results may not apply; the behavior of the methods in this setting is investigated numerically in Section 4.2.3 for the 1D case and Section 4.2.4 for the 2D case.

In order to solve problem (4.3), we adopt the Proximal Gauss–Newton method. As discussed in Example 1.12, the proximity operator of ι_C reduces to a projection onto C. Setting $H(x_k) := F'(x_k)^\top F'(x_k)$, each iteration can then be split into

two steps:

$$w_k = x_k - H(x_k)^{-1} F'(x_k)^\top F(x_k), \tag{4.5}$$

$$x_{k+1} = P_C^{H(x_k)}(w_k) = \arg\min_{v \in C} \ \frac{1}{2}\|v - w_k\|^2_{H(x_k)} = \arg\min_{v \in C} \ \frac{1}{2}\big\|F'(x_k)(v - w_k)\big\|^2. \tag{4.6}$$

The first step is a standard Gauss–Newton step, which requires the solution of a linear system with matrix $H(x_k)$; the second one is a projection onto C, taken in the metric induced by $H(x_k)$.

The scaled projection $P_C^{H(x_k)}(w_k)$ is the solution of the constrained quadratic problem

$$P_C^{H(x_k)}(w_k) = \arg\min_{v \in C} \frac{1}{2}\big\|F'(x_k)(v - w_k)\big\|^2. \tag{4.7}$$

To compute this projection, we use the projected gradient iteration introduced in Example 1.16,

$$v_{\ell+1,k} = P_C\big(v_{\ell,k} - \sigma_k H(x_k)(v_{\ell,k} - w_k)\big), \qquad v_{0,k} = w_k, \tag{4.8}$$

where P_C denotes the Euclidean projection onto C. The initialization $v_{0,k} = w_k$ serves as a *warm restart* strategy: by using the current unconstrained step as the initial guess, we accelerate the convergence of the inner iterative procedure. Notably, if the unconstrained step is already feasible (i.e., $w_k \in C$), this initialization allows the method to converge in a single step, since $v_{1,k} = P_C(w_k - 0) = w_k$. For $\sigma_k \in \big(0, 2/\lambda_{\max}(H(x_k))\big)$, the sequence $\{v_{\ell,k}\}_{\ell \in \mathbb{N}}$ converges to $P_C^{H(x_k)}(w_k)$ [4]; we choose $\sigma_k = 1/\lambda_{\max}(H(x_k))$.

The inner iterations are stopped as soon as $\|v_{\ell+1,k} - v_{\ell,k}\| < 10^{-11}$, or after a maximum number $\ell_{\max}$ of iterations, so that the scaled projection is computed only approximately. To reduce the cost of the power method, we use a warm restart: at each outer iteration it is initialized with the eigenvector estimate obtained at the previous one.

In this setting, the Proximal Gauss–Newton method provides a natural framework for combining the nonlinear least-squares structure with the convex constraints, while the multilevel approach aims to reduce the computational cost associated with fine discretizations. We begin by describing our problem and its formulation on the coarse level.

4.1.1.1 Coarse model

The coarse approximation of the objective function arises from a coarser discretization of the same continuous problem. We begin by describing the problem at the coarse level. Let $n_h := n$ denote the number of interior degrees of freedom

of the fine-grid discretization. We define the coarse level by halving the number of grid points, setting

$$n_H = \left\lfloor \frac{n_h}{2} \right\rfloor.$$

We choose to consider a coarse model with $J^H \equiv 0$, that is, we omit the box constraint at the coarse level, placing ourselves in the setting of the smooth coarse model discussed in Section 3.1. The corresponding problem reads

$$\min_{\mathbf{x}_H \in \mathbb{R}^{n_H}} \left\| A_H \mathbf{x}_H + e^{\mathbf{x}_H} - \mathbf{z}_H \right\|^2, \tag{4.9}$$

where A_H and $\mathbf{z}_H$ are the coarse-grid counterparts of A and $\mathbf{z}$. In practice, A_H is constructed using a Galerkin approximation [5], that is, the restriction/projection of A onto the coarse grid. Recall from Definition 3.1 that R denotes the restriction operator, sending information from the fine grid to the coarse grid, and P denotes the prolongation operator, sending information from the coarse grid back to the fine grid. The Galerkin approximation of A then reads

$$A_H = RAP, \tag{4.10}$$

while the coarse data vector is obtained by restricting the fine one,

$$\mathbf{z}_H = R\,\mathbf{z}. \tag{4.11}$$

The solution obtained at the coarse level is then used to provide information for the problem at a finer discretization. In particular, we introduce suitable restriction and prolongation operators to transfer information between consecutive levels.

4.1.1.2 Transfer operators

For the one-dimensional problem considered here, we choose the prolongation operator $P\colon \mathbb{R}^{n_H} \to \mathbb{R}^{n_h}$ as the standard linear interpolation operator from the coarse grid described above to the fine grid. Each coarse-grid value is directly copied to the corresponding fine-grid node, while the values at the intermediate fine-grid nodes are obtained by averaging the two neighboring coarse-grid values, the value at the left boundary being zero. Explicitly,

$$P = \frac{1}{2} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 2 & 0 & \cdots & 0 \\ 1 & 1 & \ddots & \vdots \\ 0 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 1 & 1 \\ 0 & \cdots & 0 & 2 \end{bmatrix} \in \mathbb{R}^{n_h \times n_H}. \tag{4.12}$$

Since the last fine-grid node coincides with the last coarse-grid node, the last row of P simply copies the corresponding coarse value. Let $v^H \in \mathbb{R}^{n_H}$ denote a generic vector defined on the coarse grid, and let $v^h \in \mathbb{R}^{n_h}$ denote its prolongation onto the fine grid. Explicitly, the prolongation operator acts component-wise as follows:

$$v^h_{2j} = v^H_j, \qquad v^h_{2j-1} = \tfrac{1}{2}\big(v^H_{j-1} + v^H_j\big), \qquad j = 1, \ldots, n_H, \tag{4.13}$$

with the convention $v^H_0 = 0$. This is the classical prolongation stencil used in multigrid methods [5]. The basic idea is illustrated in Figure 4.1: the values defined on the coarse grid are interpolated onto the fine grid. In particular, the values at the fine-grid points that coincide with coarse-grid points are preserved, while the values at the newly introduced points are obtained by linear interpolation between their two neighboring coarse-grid values.

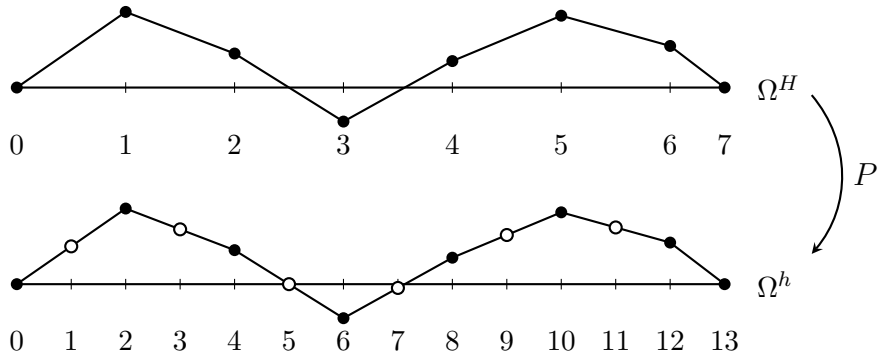


Figure 4.1: Illustration of the prolongation operator P on a one-dimensional grid with $n_h = 12$ interior fine-grid nodes and $n_H = 6$ interior coarse-grid nodes.

Consistently with Definition 3.1, the restriction operator $R\colon \mathbb{R}^{n_h} \to \mathbb{R}^{n_H}$ is chosen as

$$R = \frac{1}{2}P^\top,$$

so that $P = 2R^\top$, i.e. R and P are CIT operators with $\xi = 2$. Component-wise, each coarse-grid value is a weighted average of the neighboring fine-grid values:

$$(Rv^h)_j = \frac{1}{4}v^h_{2j-1} + \frac{1}{2}v^h_{2j} + \frac{1}{4}v^h_{2j+1}, \qquad j = 1, \ldots, n_H, \tag{4.14}$$

with the convention $v^h_{n_h+1} := 0$ when n_h is even. With the coarse level and the transfer operators defined, we can now apply the methods to the model problem.

4.1.2 Numerical experiments in 1D

We begin with a problem whose exact solution is known, so that the methods can be validated against a ground truth. We prescribe the analytical function

$$u^\star(y) = 4.5\sin(\pi y) + 2.5\sin(3\pi y), \tag{4.15}$$

which satisfies $u^\star(0) = u^\star(1) = 0$. Denoting by $\mathbf{u}^\star$ the vector of its samples at the grid points $y_i = ih$, $i = 1, \ldots, n$, we set $\mathbf{z} := A\mathbf{u}^\star + e^{\mathbf{u}^\star}$. In this way, $\mathbf{u}^\star$ is the exact solution of the discrete problem, i.e. $A\mathbf{u}^\star + e^{\mathbf{u}^\star} = \mathbf{z}$. Since the unknowns approximate the values of u at the grid points, in the experiments we denote the iterates by u_k and the initial guess by u_0, and measure the error as $\|u_k - u^\star\|_2$.

We set up the numerical experiment with the following specifications:

- **Constraint set:** we look for a solution whose values lie between -10 and 10, that is, $-10 \leq u(y) \leq 10$ for all y in the domain. After discretization, this condition becomes the uniform box constraint $C = [-10, 10]^n$ on the vector of unknowns.
- **Discretization hierarchy:** we employ a two-level framework comprising a fine grid (h) and a coarse grid (H).
- **Stopping criterion:** the iterative process terminates as soon as $\|u_k - u_{k-1}\|_2 < \varepsilon$, with a prescribed tolerance $\varepsilon = 10^{-11}$.
- **Coarse iterations:** at each V-cycle, we perform $m_c = 3$ iterations on the coarse level.

4.1.2.1 Numerical results in one dimension

We use the multilevel scheme summarized in Algorithm 5. Since the coarse model is smooth, we first minimize it by gradient descent. Figure 4.2 illustrates the algorithmic framework adopted in our implementation. In particular, the schematic depicts three consecutive V-cycles, detailing the interaction between the two discretization levels:

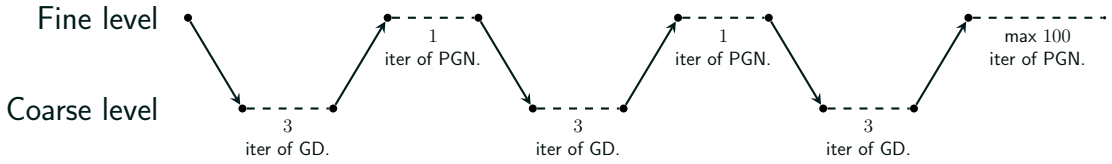


Figure 4.2: Schematic representation of the two-level multilevel PGN algorithm alternating coarse-grid corrections and fine-level smoothing cycles.

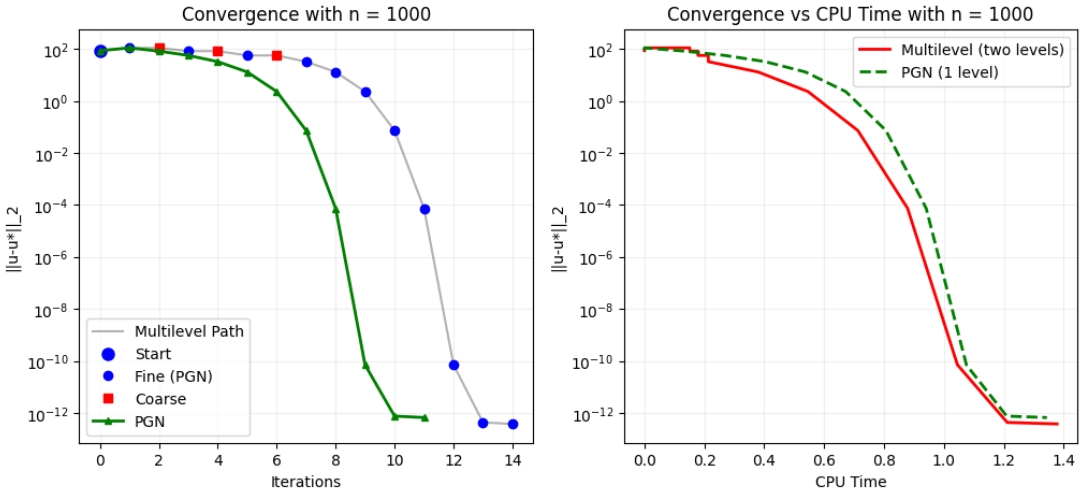


Figure 4.3: Convergence behavior for $n = 1000$, tracking the fine-level (PGN) and coarse-level steps along the multilevel path compared to standard PGN.

Remark 4.1. *In the left panel of Figure 4.3, the error is plotted against the number of steps, and each marker corresponds to one step of the corresponding method:*

- green triangles*: one iteration of standard (single-level) PGN on the fine problem;*
- blue dots*: one fine-level step of the multilevel method, that is, one PGN iteration on the fine problem, with n_h unknowns;*
- red squares*: one coarse correction of the multilevel method, which collects the m_c iterations performed on the coarse model, with $n_H \approx n_h/2$ unknowns. The plotted value is the error of the fine iterate obtained when the coarse correction is brought back to the fine level, through the prolongation.*

Steps represented by different colors have a different computational cost. Blue dots and green triangles correspond to the same operation, a PGN iteration on the fine problem, while each red square involves m_c iterations on a problem of roughly half the size. Therefore, the horizontal axis does not measure the computational effort, which is instead reported in the right panel in terms of CPU time.

Figure 4.3 shows that, in this example, the advantage provided by the multilevel framework is rather limited. Motivated by this observation, we now explore replacing the first-order gradient descent step with a Gauss–Newton solver at the coarse level, aiming to obtain a more effective coarse-scale correction.

Figure 4.4 outlines the updated two-level architecture: each cycle computes 3 iterations of the Gauss–Newton (GN) method on the coarse level to capture

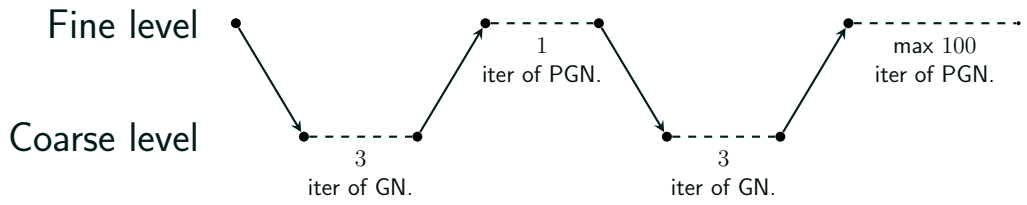


Figure 4.4: Schematic of the two-level V-cycle strategy employing coarse Gauss–Newton steps.

curvature information, followed by 1 fine-level Proximal Gauss–Newton (PGN) smoothing step to re-establish feasibility on the original box constraints.

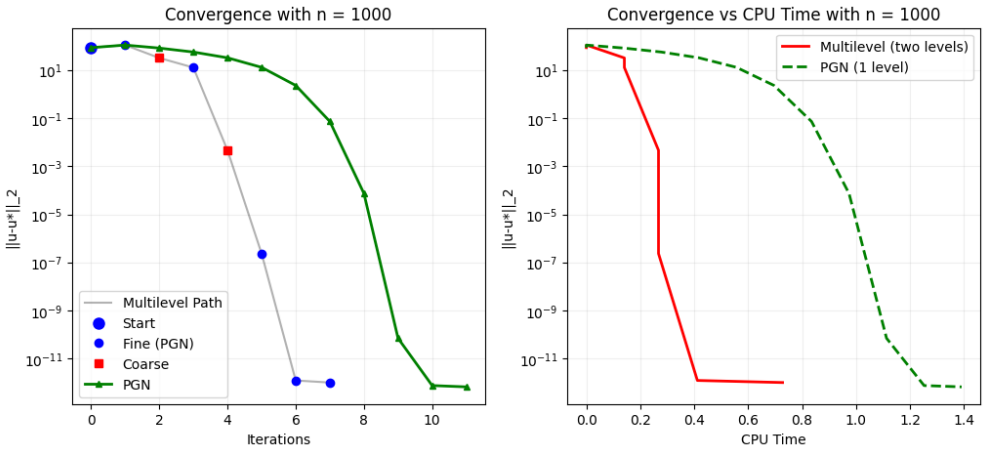


Figure 4.5: Comparison between the standard PGN method and the multilevel framework using GN at the coarse level.

Figure 4.5 shows the results obtained when using the Gauss–Newton method, instead of gradient descent, to minimize the coarse model. In this case, the advantage of the multilevel framework becomes evident: as shown in the right panel, the multilevel scheme (solid red line) reaches an accuracy of order 10^{-7} in approximately 20% of the CPU time required by the single-level Proximal Gauss–Newton method (dashed green line) to attain the same accuracy. This confirms that the choice of the coarse-level solver plays a crucial role in the overall effectiveness of the multilevel approach: while gradient descent alone was not sufficient to yield a significant speed-up, replacing it with the Gauss–Newton method allows the coarse correction to capture the relevant features of the fine-level problem much more efficiently.

We now test a larger system with $n = 3000$ to compare the standard single-level PGN against the multilevel framework using two different coarse solvers: Gradient Descent (GD) and Gauss–Newton (GN) in Figure 4.6.

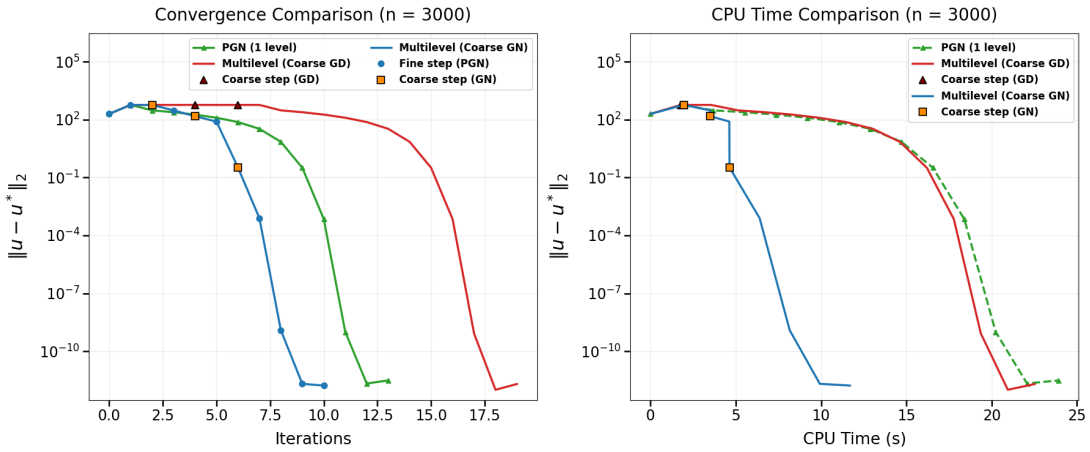


Figure 4.6: Comparison of standard PGN (1 level), multilevel with coarse GD, and multilevel with coarse GN for the coarse level.

The results clearly demonstrate the advantage of incorporating second-order information at the coarse level. In particular, we observe a significant computational advantage in intermediate accuracy regimes: to reduce the error to $\|u - u_*\|_2 \approx 10^{-7}$, the multilevel scheme with GN on the coarse grid requires less than 8 s, whereas the GD variant requires about 18 s and single-level PGN about 19 s. This advantage is mainly due to the second coarse GN correction, which reduces the error by about three orders of magnitude in a single step, from roughly 10^2 to 10^{-1}. Interestingly, although the GD variant performs more iterations than single-level PGN, it is still slightly faster in terms of CPU time, since part of the work is carried out on the cheaper coarse level. However, the advantage of the multilevel scheme with a GD coarse solver is much less evident than with GN.

4.1.3 2D problem

In this section, we present our problem of interest in two dimensions: a nonlinear elliptic PDE. Let $\Omega \subset \mathbb{R}^2$ be an open, bounded subset, and consider the following boundary value problem:

$$\begin{cases} -\Delta u(y) + e^{u(y)} = f(y) & \text{in } \Omega, \\ u(y) = 0 & \text{on } \partial\Omega, \end{cases} \tag{4.16}$$

where $u : \Omega \to \mathbb{R}$ is a twice-differentiable function, and $f : \Omega \to \mathbb{R}$ is the source term.

By replacing the continuous derivatives with second-order finite differences, we obtain the discretized version of Problem (4.16) on a uniform grid. Let the grid points be defined as

$$(y_{1,i}, y_{2,j}) = (ih, jh), \qquad h := \frac{1}{n+1}, \qquad i, j = 0, \dots, n+1,$$

and denote by $u_{i,j} \approx u(y_{1,i}, y_{2,j})$ the discrete approximation of the solution at each node. Discretizing the Laplacian by the standard five-point stencil, the discretized problem reads

$$\frac{-u_{i-1,j} - u_{i+1,j} + 4u_{i,j} - u_{i,j-1} - u_{i,j+1}}{h^2} + e^{u_{i,j}} = f_{i,j}, \qquad i,j = 1,\dots,n, \quad (4.17)$$

where $u_{0,j} = u_{n+1,j} = u_{i,0} = u_{i,n+1} = 0$ whenever a boundary node appears in the stencil.

Collecting the unknowns $u_{i,j}$, for $i,j = 1,\dots,n$, in a vector $\mathbf{x} \in \mathbb{R}^N$, with $N = n^2$, and similarly the values $f_{i,j}$ in a vector $\mathbf{z} \in \mathbb{R}^N$, system (4.17) can be written compactly as $A\mathbf{x} + e^{\mathbf{x}} = \mathbf{z}$, where $A \in \mathbb{R}^{N\times N}$ is the matrix of discrete Laplacian. Imposing in addition the box constraint $\mathbf{x} \in C$, we look for the vector that minimizes the residual of this system over C, which leads to the optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} \ \|A\mathbf{x} + e^{\mathbf{x}} - \mathbf{z}\|^2 + \iota_C(\mathbf{x}). \quad (4.18)$$

This problem has exactly the same structure as the one-dimensional problem (4.3). Consequently, the methods and the multilevel scheme of Algorithm 5 carry over to the two-dimensional case without modifications: the Proximal Gauss–Newton method is applied on the fine level, and the coarse problem is defined as in the one-dimensional case, the only difference being the transfer operators.

4.1.3.1 Transfer operators

In two dimensions, the prolongation operator is constructed as the Kronecker product of the one-dimensional prolongation operator with itself,

$$P = P_{1D} \otimes P_{1D}, \quad (4.19)$$

where P_{1D} denotes the one-dimensional prolongation operator in (4.12). Explicitly, letting $v^h = Pv^H$ and indexing the nodes by their grid coordinates, the components of v^h are given, for $i,j = 1,\dots,n_H$, by

$$\begin{aligned} v^h_{2i,2j} &= v^H_{i,j}, \\ v^h_{2i-1,2j} &= \tfrac{1}{2}\left(v^H_{i-1,j} + v^H_{i,j}\right), \\ v^h_{2i,2j-1} &= \tfrac{1}{2}\left(v^H_{i,j-1} + v^H_{i,j}\right), \\ v^h_{2i-1,2j-1} &= \tfrac{1}{4}\left(v^H_{i-1,j-1} + v^H_{i,j-1} + v^H_{i-1,j} + v^H_{i,j}\right), \end{aligned} \quad (4.20)$$

with the convention $v^H_{0,j} = v^H_{i,0} = 0$, given by the homogeneous boundary conditions. That is, the values at the fine-grid nodes that coincide with coarse-grid nodes are copied, while the other values are obtained by bilinear interpolation of the neighboring coarse-grid values.

The restriction operator is then defined, as in the one-dimensional case, as a normalized transpose of P:

$$R = \frac{1}{4}P^\top = R_{1D} \otimes R_{1D}, \tag{4.21}$$

since $R_{1D} = \frac{1}{2}P_{1D}^\top$. Therefore $P = 4R^\top$, so that (R, P) is a pair of CIT operators, in the sense of Definition 3.1, with $\xi = 4$.

4.1.4 Numerical experiments in 2D

To test our solver, we prescribe an analytical function on $\Omega = (0,1) \times (0,1)$,

$$u^\star(y_1, y_2) = 4.5 \sin(\pi y_1) \sin(\pi y_2) + 2.5 \sin(3\pi y_1) \sin(3\pi y_2),$$

which vanishes on $\partial\Omega$. As in the one-dimensional case, denoting by $\mathbf{u}^\star$ the vector of its samples at the grid points, we set $\mathbf{z} := A\mathbf{u}^\star + e^{\mathbf{u}^\star}$, so that $\mathbf{u}^\star$ is the exact solution of the discrete problem. The numerical experiment is then set up as follows:

- **Constraint set:** we consider the uniform box constraint $C = [-10, 10]^N$, with $N = n^2$.
- **Grid hierarchy:** as in the one-dimensional case, we utilize two spatial discretization levels (a fine and a coarse one).
- **Algorithmic setup:** the stopping criterion and the definition of the coarse model are identical to those discussed in the 1D problem, and at each V-cycle we perform $m_c = 2$ iterations on the coarse level.

4.1.4.1 Numerical results in two dimensions

We begin by testing our algorithm with $n = 700$, so that the total number of degrees of freedom is $N = n^2 = 700^2$. Within the multilevel V-cycle framework, we perform 2 iterations of the Gauss–Newton method at the coarse level.

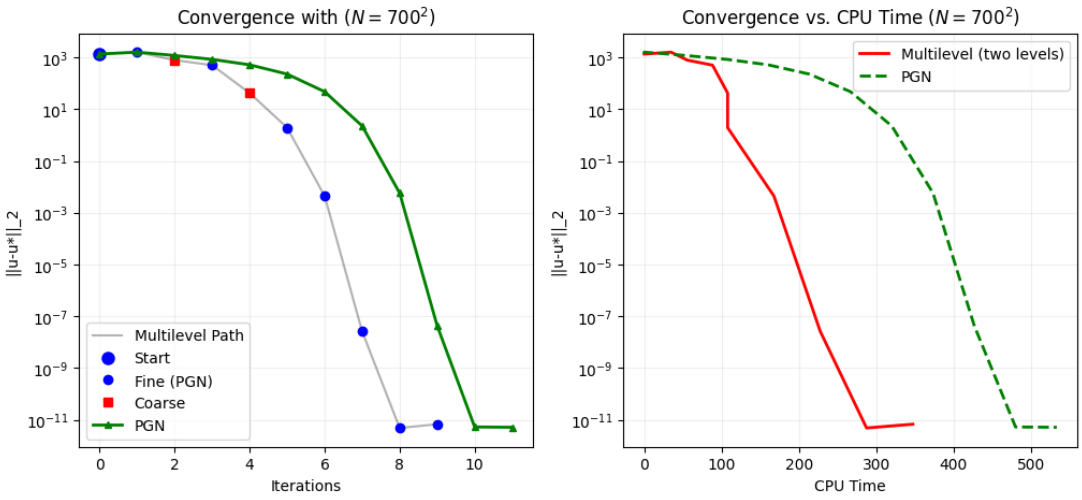


Figure 4.7: Convergence and CPU time comparison between the two-level multilevel solver and standard PGN for $N = 700^2$.

Figure 4.7 compares the two-level multilevel method with standard PGN on a 700×700 grid ($N = 700^2$). In terms of iterations (left panel), the coarse corrections accelerate convergence: the multilevel method reaches an accuracy of order 10^{-11} in about 8 iterations, compared to 10 for standard PGN. In terms of CPU time (right panel), this results in a significant reduction of the computational cost: to reach an error of order 10^{-5}, the multilevel method requires about 200 seconds, roughly half the time needed by standard PGN, which requires about 400 seconds. In the examples considered so far, we adopted a sufficiently large box constraint, chosen so that the exact solution $u^\star$ lies well within its interior. As a consequence, the box constraint at the coarse level was rarely, if ever, active, since the iterates naturally remained feasible without the constraint playing an effective role. This likely explains why using a non-smooth coarse model, with $J^H = \iota_{C_H}$, did not provide a significant advantage over the smooth coarse model with $J^H \equiv 0$ discussed above.

Motivated by this observation, we now consider a smaller box constraint, chosen so that the solution lies closer to its boundary, in order to assess whether the non-smooth coarse model becomes more advantageous when the constraint is genuinely active.

4.2 Non-smooth coarse problem

In this section, we consider a smaller box constraint. In this case, the bounds are expected to be active on a significant part of the domain, so the constraint can no longer be neglected at the coarse level, and a projection must be introduced there as well. To make the projection at the coarse level meaningful, however, we cannot

simply impose at the coarse level the same bounds that define C: the correction computed at the coarse level is subsequently prolongated back to the fine grid and used to update the fine-level iterate, and a coarse iterate satisfying the original bounds may still produce, once prolongated, a fine-level update that leaves the feasible set. It is therefore necessary to define a suitable coarse constraint set C_H, which accounts for the effect of the prolongation and ensures that the resulting fine-level iterate remains in C. This is precisely the issue addressed by Gratton et al. [15] in the context of recursive trust-region methods for multilevel bound-constrained optimization.

4.2.1 Construction of the coarse constraint set

Rather than imposing the original bounds at every level, they define the coarse constraints so that any coarse correction, once prolongated, keeps the fine-level iterate feasible. They construct, at each coarser level $i-1$, a bound-constrained domain $\mathcal{F}_{i-1}$ defined recursively from the corresponding domain $\mathcal{F}_i$ at the finer level. The key idea is to account for the current distance of the fine-level iterate to the boundary of C, appropriately rescaled through the restriction operator: intuitively, the coarse feasible region is shrunk in proportion to how close the fine-level iterate already is to the boundary, so that any correction computed at the coarse level, once prolongated back to the fine level, is guaranteed to remain feasible. This recursive construction ensures that feasibility is a property propagated consistently across the entire hierarchy of levels, rather than being imposed independently and somewhat arbitrarily at each one, avoiding precisely the risk we outlined above.

We now describe how to define the domains $\mathcal{F}_{i-1}$ at the coarse level. Our goal is to guarantee that

$$x_{i,k} + P_i s_{i-1} \in \mathcal{F}_i \qquad \text{for all } x_{i-1,0} + s_{i-1} \in \mathcal{F}_{i-1}, \tag{4.22}$$

so that any coarse correction step, once prolongated back to the fine level, is guaranteed to remain feasible.

Write $\mathcal{F}_i = [a_i, b_i]$ and $\mathcal{F}_{i-1} = [a_{i-1}, b_{i-1}]$, and recall that $x_{i-1,0} = R_i x_{i,k}$. Since the prolongation operator P_i has nonnegative entries, it preserves componentwise ordering: if $\alpha \le s_{i-1} \le \beta$, then $P_i\alpha \le P_i s_{i-1} \le P_i\beta$. In particular, over all $s_{i-1} \in [a_{i-1} - x_{i-1,0},\ b_{i-1} - x_{i-1,0}]$, the extremal values of $P_i s_{i-1}$ are attained at the bounds themselves, namely $P_i(a_{i-1} - x_{i-1,0})$ and $P_i(b_{i-1} - x_{i-1,0})$. Condition (4.22) is therefore satisfied as soon as

$$a_i - x_{i,k} \;\le\; P_i\,(a_{i-1} - x_{i-1,0}) \qquad \text{and} \qquad P_i\,(b_{i-1} - x_{i-1,0}) \;\le\; b_i - x_{i,k}. \tag{4.23}$$

A simple and natural choice of $\mathcal{F}_{i-1}$ satisfying (4.23) is obtained by taking the most restrictive bound over all fine-level indices, rescaled by $\|P_i\|_\infty$. The bound

vectors a_{i-1} and b_{i-1} are thus defined component-wise by

$$[a_{i-1}]_j = [x_{i-1,0}]_j + \frac{1}{\|P_i\|_\infty} \max_{t=1,\dots,n_i} [a_i - x_{i,k}]_t \tag{4.24}$$

and

$$[b_{i-1}]_j = [x_{i-1,0}]_j + \frac{1}{\|P_i\|_\infty} \min_{t=1,\dots,n_i} [b_i - x_{i,k}]_t, \tag{4.25}$$

for $j = 1, \dots, n_{i-1}$. This construction, in the spirit of the one proposed by Gratton et al. [15], defines the coarse feasible domain $\mathcal{F}_{i-1}$ recursively from $\mathcal{F}_i$ and the current fine-level iterate $x_{i,k}$: the coarse box is shrunk exactly by the amount needed to guarantee that no coarse correction, however large within $\mathcal{F}_{i-1}$, can cause the corresponding fine-level update to leave $\mathcal{F}_i$.

We rely on this specific choice because, as shown in [15, Lemma 4.3], these definitions enforce the inclusion

$$x_{i,k} + P_i(x_{i-1} - x_{i-1,0}) \in \mathcal{F}_i \quad \text{for all } x_{i-1} \in \mathcal{F}_{i-1}. \tag{4.26}$$

As a consequence, we have $x_{i,k} \in \mathcal{F}_i$ for all levels i and all iterations $k \geq 0$, which ensures that the iterates remain feasible throughout the entire procedure.

With a tighter box, an additional safeguard is required to control the length of the step. In the previous experiments, the constraint was essentially inactive and the unit step was always accepted, so no step-size selection was needed. When the bounds become active, however, the full step along the direction provided by the method is no longer guaranteed to decrease the objective function: it may overshoot, or move the iterate into a region where the linearized model is no longer accurate. For this reason, we equip the Proximal Gauss–Newton method with a backtracking line search. Given the current iterate x_k, the proximal point $v_k = P_C^{H(x_k)}(w_k)$ and the corresponding direction $d_k = v_k - x_k$, we start from $\alpha = 1$ and reduce it by a factor $\tau = 0.5$ until

$$\Phi(x_k + \alpha d_k) < \Phi(x_k). \tag{4.27}$$

The same procedure is also applied to the prolongated coarse correction $d_k = P s_H^\star$, which is not guaranteed to reduce the fine-level objective at full step length; in this case a strict decrease is required.

Having adapted the multilevel framework to a restricted box domain, through the construction of the coarse constraint set and the backtracking safeguard, we are now ready to test it in the setting where the box constraint is active.

4.2.2 Setting with an active box constraint

We now present our results using a smaller box constraint. Unlike the previous experiments, however, we no longer have access to the exact solution against which to measure the error: since the box constraint is now active, the constrained

solution of Problem (4.3) generally differs from the manufactured profile $\hat{u}$ used to construct the data $\mathbf{z}$, which was designed to solve the unconstrained problem. Consequently, in this setting we can no longer monitor convergence through the error with respect to the exact solution. Instead, we monitor the objective function value $\Phi(u_k) = \frac{1}{2}\|F(u_k)\|^2$, where the indicator term vanishes since all iterates are feasible, while the increment $\|u_{k+1} - u_k\|$ is used in the stopping criterion.

Our aim, as always, is to solve

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ \|A\mathbf{x} + e^{\mathbf{x}} - \mathbf{z}\|^2 + \iota_C(\mathbf{x}), \tag{4.28}$$

now with a smaller box C.

4.2.3 Numerical results in 1D

As always, we begin by examining the one-dimensional case. To systematically generate the right-hand side vector $\mathbf{z}$ (the data of our problem), we first define a continuous target profile $\hat{u}(y)$, evaluated on the spatial grid. Specifically, we choose the following function:

$$\hat{u}(y) = 4.5\sin(\pi y) + 2.5\sin(3\pi y).$$

The vector $\mathbf{z}$ is then explicitly constructed by applying the forward operator to this profile:

$$\mathbf{z} = A\hat{u} + e^{\hat{u}}.$$

In an unconstrained scenario, $\hat{u}$ would clearly be the global minimizer of the objective function. However, by deliberately setting a tighter box constraint C such that $\hat{u} \notin C$, the target profile becomes infeasible.

Consequently, the constrained minimizer of the problem is not known analytically. This setup is designed to test the methods in a setting where the box constraint is active on a significant portion of the domain, and allows us to assess the behavior of ML-PGN in the presence of active constraints. For this experiment, we impose a tighter box constraint on the solution, requiring its values to lie between -1 and 5. All other algorithmic parameters and grid specifications remain identical to those described for the one-dimensional case with a smooth coarse model, and at each V-cycle we perform $m_c = 2$ iterations on the coarse level. We decided to use PGN also for the coarse model, with the box constraint defined in Equations (4.24)–(4.25).

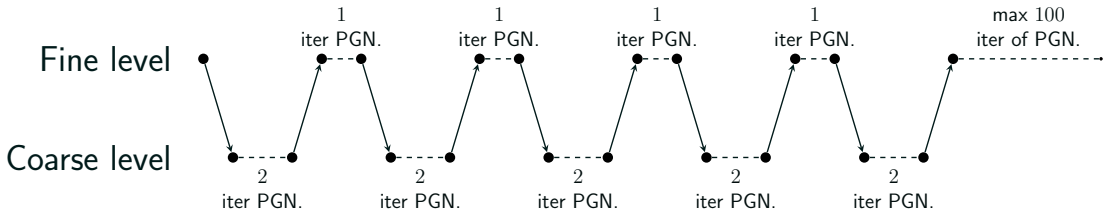


Figure 4.8: Schematic of the two-level strategy with five V-cycles, in which the coarse correction is computed by Proximal Gauss–Newton iterations.

We now present our results, considering $n = 1000$. Since we do not have access to the exact solution in this setting, we instead plot the objective function value $\Phi(u_k) = \frac{1}{2}\|F(u_k)\|^2$, as a proxy for convergence. The initial point u_0 is chosen at random.

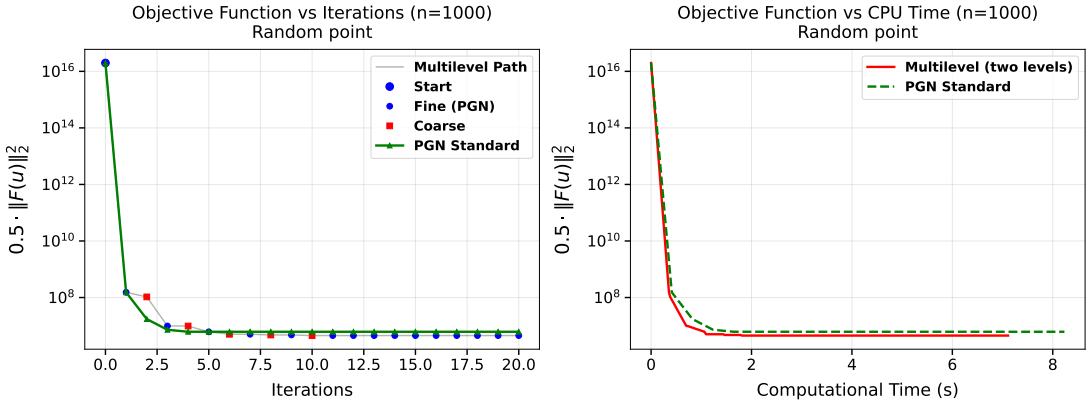


Figure 4.9: Objective function decrease for a random initial vector, comparing the proposed multilevel framework and the standard PGN method.

Figure 4.9 shows the results obtained for $n = 1000$, comparing the multilevel scheme against the standard, single-level PGN method. We monitor convergence through the value of the objective function, both as a function of the number of iterations (left panel) and of computational time (right panel).

Two distinct advantages of the multilevel scheme emerge from this comparison. First, as shown in the right panel, the multilevel method reaches its final objective value in a substantially shorter computational time than standard PGN. Second, and more notably, the multilevel scheme does not merely converge faster to the same value: as shown in the left panel, it attains a strictly lower final objective value than standard PGN, namely $4.51 \cdot 10^6$ against $6.15 \cdot 10^6$. This suggests that, in this constrained setting, the coarse correction helps the fine-level iterates escape a region where standard PGN alone gets stuck.

We remark that standard PGN stagnates after very few iterations, so the stopping criterion would terminate it almost immediately. In order to compare the

methods on an equal footing, we therefore required both standard PGN and the multilevel scheme to perform at least 20 iterations. The same setting is adopted in all the following experiments.

We now consider a different initial guess, $u_0 \equiv 0$.

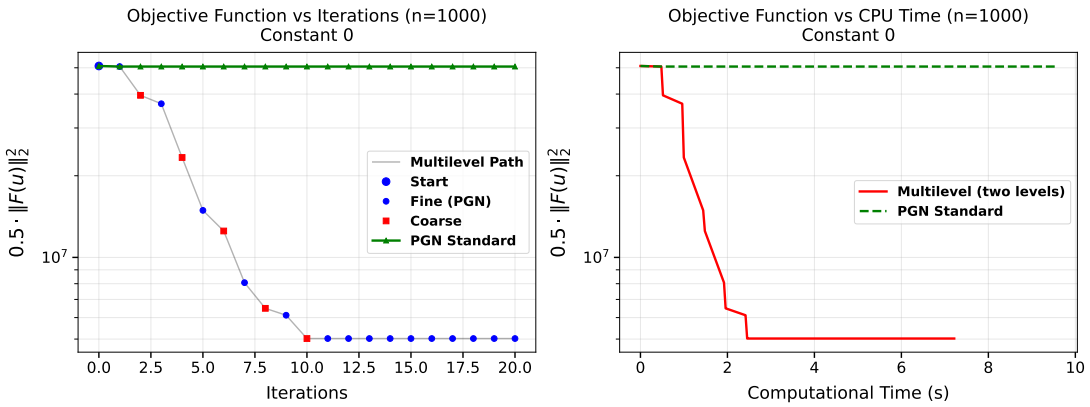


Figure 4.10: Objective function decay for the constant initial vector $u_0 = 0$, comparing the proposed multilevel framework and the standard PGN method.

The standard PGN method fails to make any noticeable progress in minimizing the objective function, stagnating immediately from the initial iteration at a value of approximately $5.05 \cdot 10^7$. In contrast, the multilevel framework effectively overcomes this stagnation: the coarse-level corrections break the trapped state and drive a rapid reduction of the objective function across one order of magnitude, eventually settling at $5.02 \cdot 10^6$.

Moreover, as illustrated in the right panel, this substantial decrease is achieved with remarkable computational efficiency: the multilevel scheme reaches its lower plateau in less than 3 seconds of CPU time, whereas standard PGN remains stalled at its initial residual throughout the entire execution time of nearly 10 seconds.

We also incorporate the Levenberg–Marquardt scheme as the fine-level smoother within the multilevel framework; in the subsequent results, we compare this approach against both the standard PGN and the single-level PGN-LM methods.

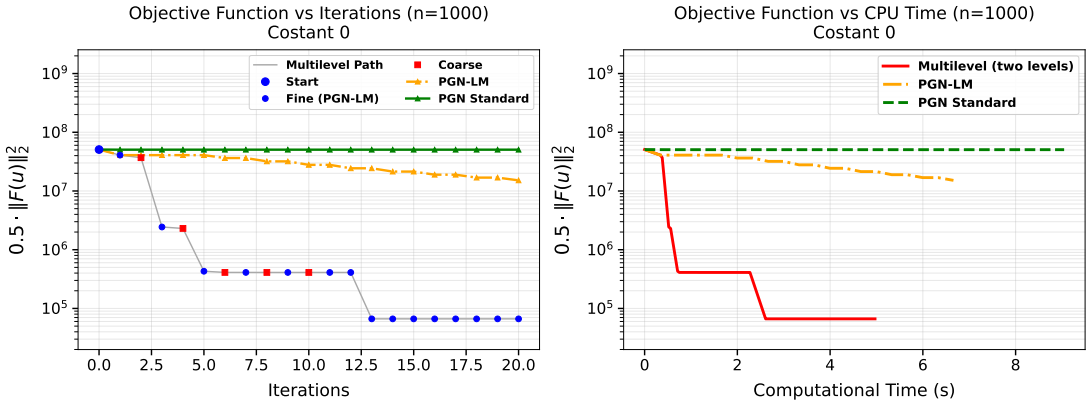


Figure 4.11: Comparison of the objective function between the multilevel scheme (with fine-level PGN-LM), single-level PGN-LM, and standard PGN.

A closer inspection of the objective function decay reveals a remarkable hierarchy in performance across the evaluated algorithms (Figure 4.11).

While the standard PGN method stagnates immediately at $\Phi \approx 5.05 \cdot 10^7$, standalone PGN-LM manages to initiate a slow descent, settling at $\Phi_{\text{last}} \approx 1.52 \cdot 10^7$ after approximately 7 seconds. We can see that integrating the Levenberg–Marquardt smoother at the fine level (ML-LM) provides an even more dramatic improvement.

Specifically, the ML-LM algorithm achieves the lowest objective function value among all configurations, settling at:

$$\Phi_{\text{last}}(\text{ML-LM}) \approx 6.65 \cdot 10^4.$$

This represents an improvement of nearly two orders of magnitude compared to standard fine-level multilevel relaxation ($5.02 \cdot 10^6$) and more than two orders of magnitude compared to single-level PGN-LM ($1.52 \cdot 10^7$). Furthermore, as illustrated in the right panel, ML-LM executes this extensive residual reduction in less than 3 seconds of CPU time.

We now present our results, considering the set $C = [-1, 4]^n$ and $n = 1000$.

The next figure compares the convergence history of standard PGN and the two-level multilevel method with the PGN-standard at the fine level for $n = 1000$, starting from the constant initialization $u_0 \equiv 1$.

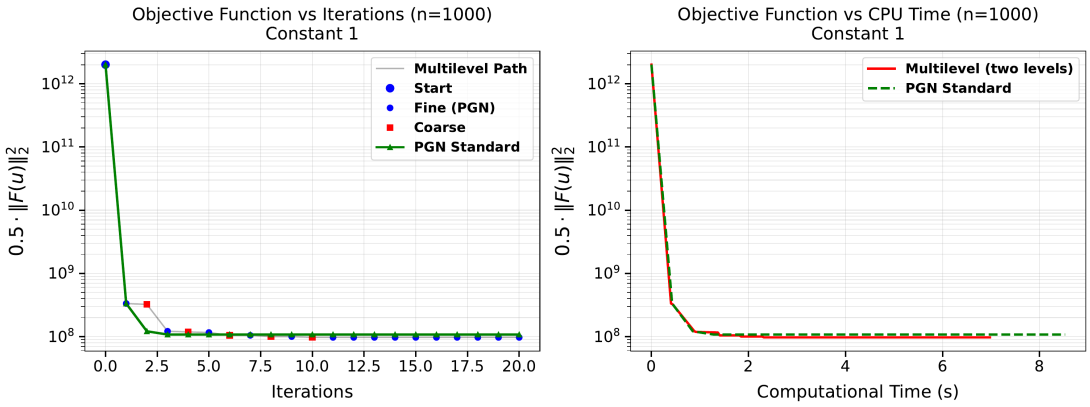


Figure 4.12: Minimization history of the objective function with $n = 1000$.

As observed in the left panel, both solvers achieve an abrupt decrease in the objective function value. The right panel shows the same reduction plotted against CPU time. During the initial sharp decrease, the two methods perform almost identically, since the fast local convergence of the Gauss–Newton step dominates at this stage. The gap opens up later: the multilevel scheme reaches its final plateau noticeably faster than standard PGN, reflecting a lower average cost per iteration once coarse corrections come into play.

Now we incorporate the Proximal Levenberg–Marquardt method as the fine-level solver within the multilevel framework.

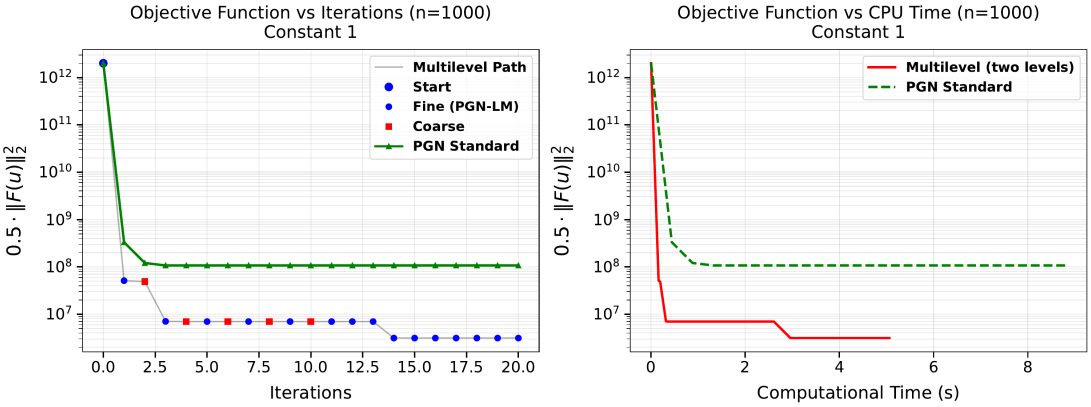


Figure 4.13: Multilevel algorithm by adopting PGN-LM at the fine level vs PGN-standard.

By substituting the standard PGN smoother with the Proximal Levenberg–Marquardt (PGN-LM) method at the fine level of the two-level framework, a substantial improvement in solution quality is achieved.

Specifically, while the multilevel scheme with standard fine-level PGN stalls at the plateau $\Phi_{\text{ML-PGN}} \approx 1.08 \cdot 10^8$, adopting PGN-LM allows the multilevel iterates

to break through this stagnation and settle at $\Phi_{\text{ML-LM}} \approx 3.13 \cdot 10^6$. This yields an objective function reduction of:

$$\Phi_{\text{ML-PGN}} - \Phi_{\text{ML-LM}} \approx 1.05 \cdot 10^8,$$

confirming that the adaptive Levenberg–Marquardt damping at the fine grid prevents premature freezing of the coordinates on the active box bounds and drives the overall multilevel hierarchy toward a significantly more accurate minimizer. We now present our numerical results in the two-dimensional setting.

4.2.4 Numerical results in 2D

We now analyze the two-dimensional case. As before, we define $\mathbf{z}$ following the same approach used in the one-dimensional case, with the target profile now given by

$$\hat{u}(y_1, y_2) = 4.5 \sin(\pi y_1) \sin(\pi y_2) + 2.5 \sin(3\pi y_1) \sin(3\pi y_2). \tag{4.29}$$

This profile is used to construct $\mathbf{z} = A\hat{u} + e^{\hat{u}}$, and a tighter box constraint C is chosen so that $\hat{u} \notin C$, making the target profile infeasible and the true constrained minimizer analytically unknown. We now impose the bounds $-1 \leq u(y_1, y_2) \leq 5$ in Ω, which, after discretization, correspond to the box $C = [-1, 5]^N$, with $N = n^2$. In this case, we compare three methods: standard PGN, ML-PGN, which uses standard PGN on both levels, and ML-LM, which uses PGN-LM on the fine level and standard PGN on the coarse level. Both multilevel variants perform five V-cycles.

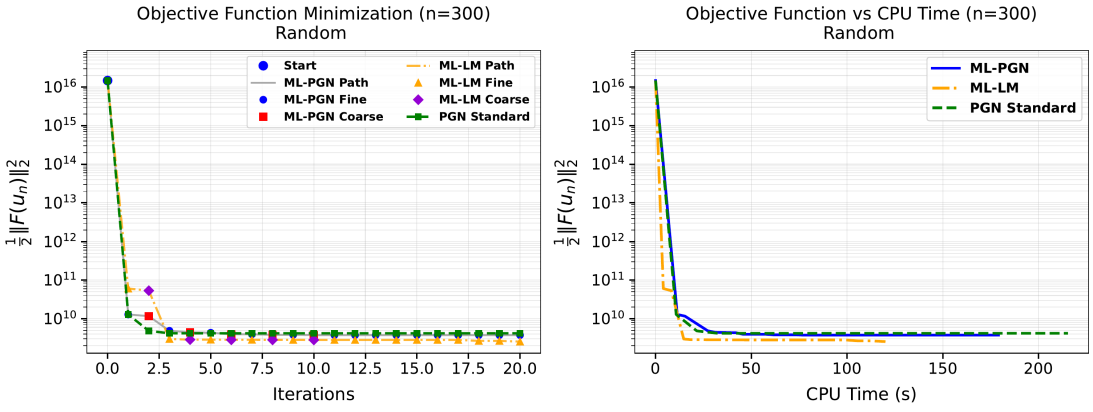


Figure 4.14: Comparison between standard PGN, multilevel PGN, and multilevel PGN with the Levenberg–Marquardt regularization.

Figure 4.14 shows the results obtained starting from a random initial point, for $n = 300$. We observe that using PGN-LM at the fine level (yellow path) is the best-performing option, both in terms of computational time and of the final objective value attained, despite exhibiting a slower initial decrease compared to

the other two variants. In particular, ML-LM reaches a final value of $2.55 \cdot 10^9$, which is lower than the $3.75 \cdot 10^9$ obtained by ML-PGN, and also better than the value reached by standard PGN ($4.19 \cdot 10^9$). Moreover, ML-LM converges to its final value in roughly 100 s, noticeably faster than both ML-PGN and standard PGN, which require significantly more time to reach a higher objective value.

We now consider the constant initial guess $u_0 \equiv 0$ and compare the four methods: standard PGN, PGN with Levenberg–Marquardt regularization (PGN-LM), the multilevel scheme with standard PGN at both the fine and the coarse level (ML-PGN), and the multilevel scheme with PGN-LM at the fine level and standard PGN at the coarse level (ML-LM). For both multilevel variants, we perform five V-cycles.

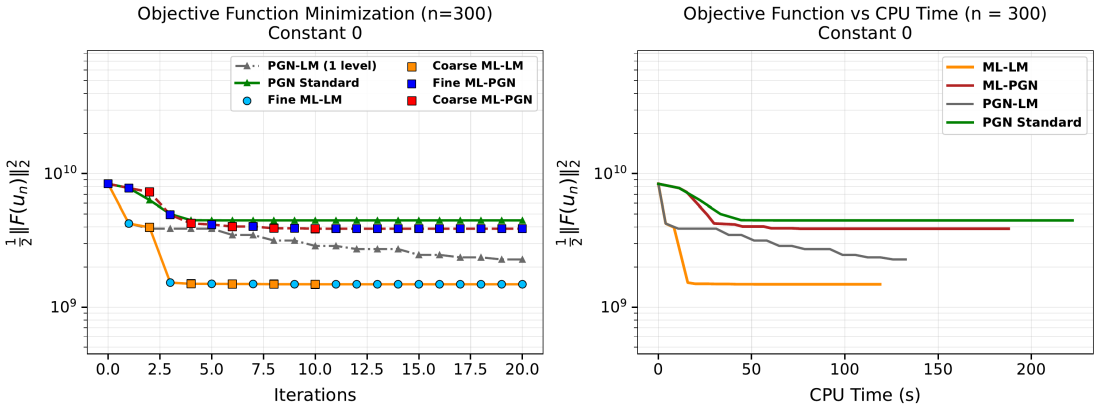


Figure 4.15: Comparison of the four methods: standard PGN, PGN-LM, multilevel PGN and multilevel PGN-LM.

In this case, the Levenberg–Marquardt regularization leads to clear improvements over both standard PGN and the multilevel scheme with standard PGN at the fine level (ML-PGN), both in terms of CPU time and of the objective function value reached. Standard PGN stagnates after about four iterations, ending at $\frac{1}{2}\|F(u_k)\|_2^2 \approx 4.48 \cdot 10^9$ after roughly 195 seconds. ML-PGN behaves similarly: the coarse corrections allow it to go slightly below the plateau of standard PGN, but it also stagnates, reaching $3.88 \cdot 10^9$. Adding the regularization changes the picture considerably. Single-level PGN-LM keeps decreasing along all the iterations and reaches $2.28 \cdot 10^9$, but the best result is obtained by ML-LM: right after the first coarse correction, the objective drops sharply to about $1.5 \cdot 10^9$.
We now start from the constant initial guess $u_0 \equiv 1$.

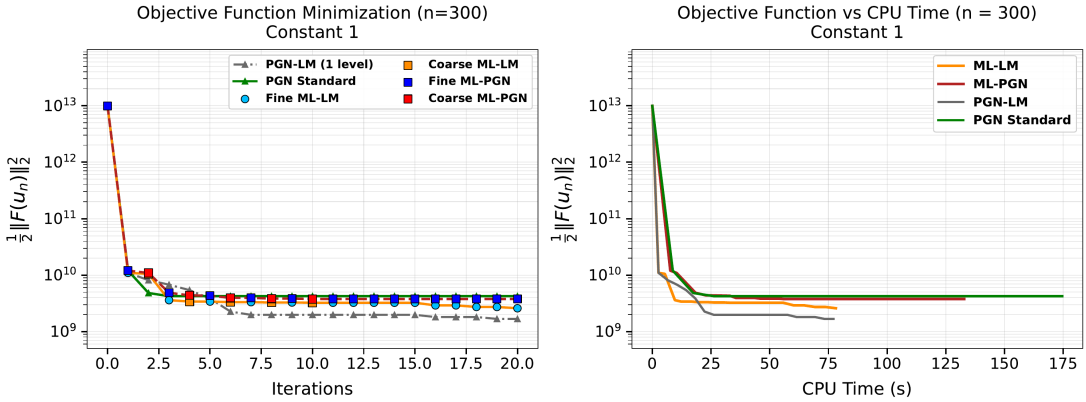


Figure 4.16: Performance comparison among standard PGN, PGN-LM, multilevel PGN, and multilevel PGN-LM in terms of convergence history and computational time.

Unlike the case $u_0 \equiv 0$, where ML-LM turned out to be the most efficient method, here the best results are obtained by single-level PGN-LM, which reaches a final value of $1.68 \cdot 10^9$, which is lower than standard PGN ($4.24 \cdot 10^9$) and ML-LM ($2.59 \cdot 10^9$). In the first iterations ML-LM actually decreases the objective faster, reaching about $3.5 \cdot 10^9$ within roughly 15 seconds, but it then remains almost stagnant, while PGN-LM keeps decreasing, with a comparable total CPU time. Standard PGN and ML-PGN stagnate at higher values, with ML-PGN reaching $3.78 \cdot 10^9$, and both require considerably more CPU time. As in the previous case, the Levenberg–Marquardt regularization improves the performance of both the single-level and the multilevel scheme. However, whether the coarse corrections are beneficial seems to depend on the initial guess: they clearly accelerate convergence when starting from $u_0 \equiv 0$, whereas for $u_0 \equiv 1$ they only provide an advantage in the very first iterations.

We now consider the tighter box constraint $-1 \leq u(y_1, y_2) \leq 2$ in Ω, which, after discretization, corresponds to $C = [-1, 2]^N$, and the constant initial guess $u_0 \equiv 1$.

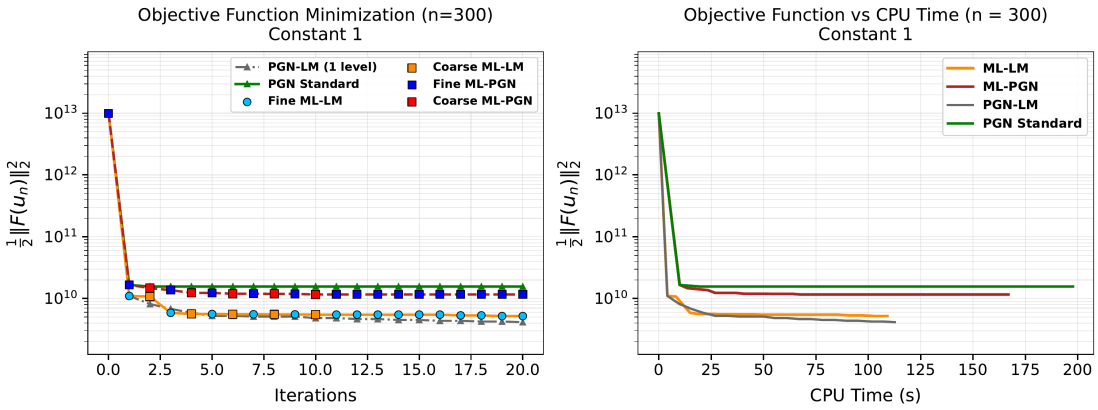


Figure 4.17: Comparison of the four methods with initial guess $u_0 \equiv 1$.

In Figure 4.17 we can see that standard PGN shows a clear stagnation. After a sharp initial decrease, from about 10^{13} to about $1.5 \cdot 10^{10}$, the objective function remains on a plateau for all the remaining iterations. In the CPU time panel, the method keeps iterating for more than 200 seconds without any further improvement.

The multilevel version with standard PGN at the fine level (ML-PGN) goes slightly below this plateau thanks to the coarse corrections, reaching about $1.1 \cdot 10^{10}$. However, the progress stops shortly afterwards. This suggests that the coarse corrections alone are not enough to overcome the limitations of the unregularized fine-level solver.

The Levenberg–Marquardt regularization allows single-level PGN-LM to avoid the stagnation of standard PGN: the objective keeps decreasing steadily and falls below $5 \cdot 10^9$ in about 120 seconds, reaching the lowest final value among the four methods.

Finally, the behavior of ML-LM is very close to that of PGN-LM. In terms of CPU time, it shows a slightly faster initial decrease, reaching values around $5.5 \cdot 10^9$. For this test case, moving to the coarse level does not seem to be beneficial: the computational overhead introduced by the coarse corrections is not compensated by a significant improvement, and PGN-LM turns out to be the most effective choice. Although it requires a slightly longer CPU time than ML-LM, the difference is only a few seconds, while PGN-LM reaches a lower final objective value.

In order to investigate a possible reason for the better performance of PGN-LM compared to standard PGN, we monitored the condition number of the matrix used in the Gauss–Newton step at each iteration.

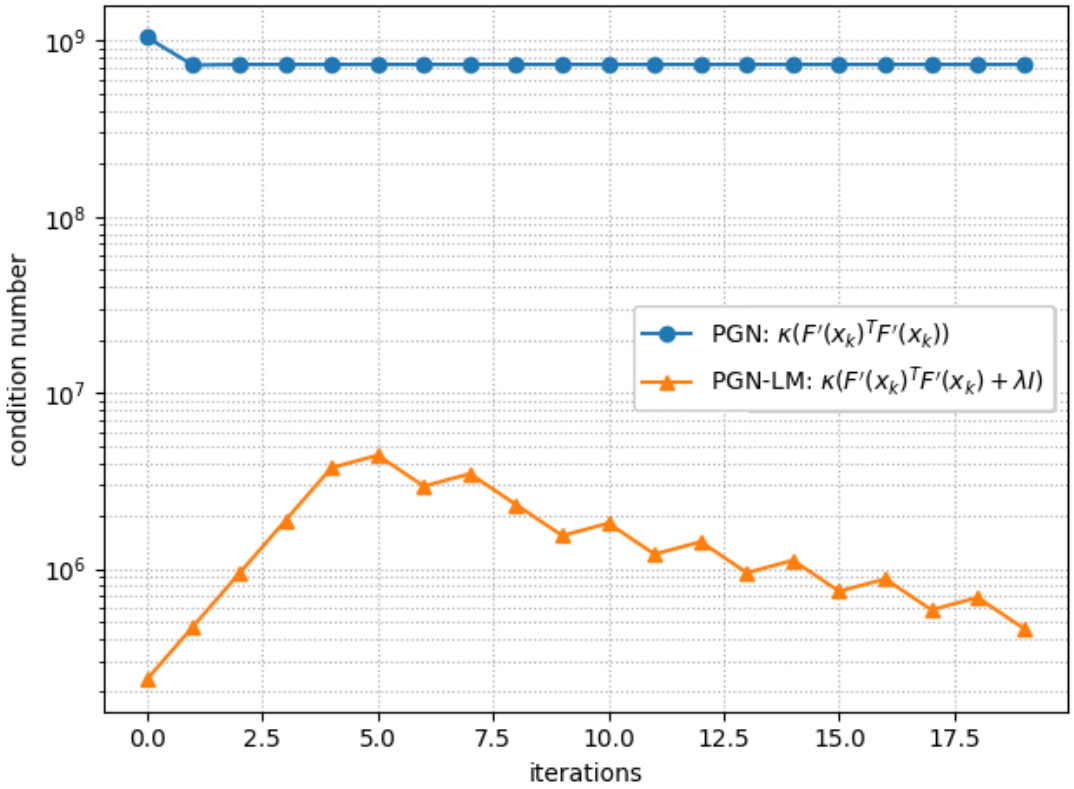


Figure 4.18: Evolution of the condition number along the iterations.

A possible factor behind the better convergence of the PGN-LM scheme may lie in the regularization of the linearized system. As reported in Figure 4.18, the matrix $F'(x_k)^\top F'(x_k)$, where $F'(x_k)$ denotes the Jacobian of F at the current iterate, is severely ill-conditioned in standard PGN, with $\kappa(F'(x_k)^\top F'(x_k)) \approx 10^9$ throughout the iterations. In contrast, the adaptive Levenberg–Marquardt parameter λ shifts the spectrum and reduces the condition number by two to three orders of magnitude, keeping $\kappa(F'(x_k)^\top F'(x_k) + \lambda I)$ between roughly 10^5 and $5 \cdot 10^6$ along the iterations. Since the convergence of the inner projected gradient iterations depends on κ, this difference could allow the regularized method to compute a more accurate step at each iteration, which might explain, at least in part, its faster convergence compared to standard PGN.

Conclusions & Outlook

In this thesis, we have studied second-order methods for composite optimization problems of the form

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2}\|F(x)\|^2 + J(x),$$

where F is a nonlinear operator and J is a convex, possibly non-smooth term. Our starting point was the Proximal Gauss–Newton (PGN) method, which, unlike first-order schemes such as Proximal Gradient Descent, exploits curvature information and achieves fast local convergence.

The first contribution of this work is a Levenberg–Marquardt regularization of PGN (PGN-LM). On the one hand, it extends the Levenberg–Marquardt method, originally designed for smooth least-squares problems, to the composite setting. On the other hand, it makes PGN applicable when the Gauss–Newton matrix is singular or ill-conditioned. For the resulting method, we have established a local linear convergence result.

The second contribution is the combination of PGN and PGN-LM with a multilevel framework, leading to the methods ML-PGN and ML-LM. When the problem arises from the discretization of a PDE, this framework exploits the hierarchy of discretizations, computing corrections on coarser, cheaper levels and transferring them to the fine level. We have discussed the construction of the coarse model, the treatment of box constraints on the coarse level, and the selection of the step length through a backtracking line search.

The numerical experiments, carried out on a box-constrained nonlinear elliptic problem in one and two dimensions, led to the following observations.

- The Levenberg–Marquardt regularization consistently improved the performance of PGN, both in terms of the objective value reached and of CPU time. Standard PGN often stagnated after a few iterations, whereas PGN-LM kept decreasing the objective. A possible explanation is the improved conditioning of the regularized system, which reduced the condition number by two to three orders of magnitude and made the inner projected gradient iterations more effective.

- In the smooth setting, using second-order information also on the coarse level proved clearly advantageous: the multilevel scheme with a Gauss–Newton

coarse solver required far less CPU time than both single-level PGN and the variant with a gradient descent coarse solver.

- In the constrained setting, the benefit of the coarse corrections turned out to depend on the test case, and in particular on the initial guess: in some cases ML-LM was the most efficient method, while in others single-level PGN-LM performed better. This shows that descending to the coarse level is not always convenient, since the computational overhead of the coarse corrections pays off only if they produce a sufficient decrease of the fine-level objective.

This work opens up several theoretical and numerical directions for future research:

- **Theoretical analysis and multigrid extensions.** A formal convergence analysis of the ML-PGN and ML-LM methods is a natural theoretical continuation of this work. Furthermore, extending the current two-level architecture to a full multigrid framework, employing complete V-cycles across more than two levels, could maximize the scalability of the algorithm on extremely fine meshes.

- **Globalization.** The convergence result for PGN-LM is local. Globalization strategies, such as line searches based on a sufficient decrease condition or trust-region techniques, could guarantee convergence from arbitrary initial guesses, which is especially relevant given the sensitivity to the initial guess observed in our experiments.

- **Applications in imaging.** A highly promising direction is the application of the proposed multilevel PGN framework to the field of imaging. Large-scale inverse problems, such as image reconstruction, denoising, and deblurring, mathematically translate to high-dimensional optimization tasks. These applications inherently possess a grid-like structure, making them an ideal testing ground to fully exploit the computational efficiency and scalability of our multilevel approach.

Bibliography

[1] S. Salzo and S. Villa, *Proximal Gradient Methods for Machine Learning and Imaging*. Springer Nature Switzerland AG, 2021.
DOI: https://doi.org/10.1007/978-3-030-86664-8_4

[2] H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137:91–129, 2013.

[3] Dennis, J. E. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall.

[4] S. Salzo and S. Villa, *Convergence analysis of a proximal Gauss-Newton method*, Computational Optimization and Applications, vol. 53, no. 2, pp. 557–589, 2012.

[5] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., Society for Industrial and Applied Mathematics, 2000.

[6] H. Calandra, S. Gratton, E. Riccietti, and X. Vasseur, *On high-order multilevel optimization strategies*, Technical report, 2019.

[7] H. Calandra, S. Gratton, E. Riccietti, and X. Vasseur. *On the approximation of the solution of partial differential equations by artificial neural networks trained by a multilevel Levenberg–Marquardt method. Technical report*, 2019. https://arxiv.org/pdf/1904.04685

[8] S. G. Nash, *A Multigrid Approach to Discretized Optimization Problems*, *Optimization Methods and Software*, vol. 14, no. 1-2, pp. 99–116, 2000.

[9] A. Javaherian and S. Holman, *A Multi-Grid Iterative Method for Photoacoustic Tomography*, *IEEE Transactions on Medical Imaging*, pp. 696–706, 2017.

[10] S. W. Fung and Z. Wendy, *Multigrid Optimization for Large-Scale Ptychographic Phase Retrieval*, *SIAM Journal on Imaging Sciences*, vol. 13, pp. 214–233, 2020.

[11] P. Parpas, *A Multilevel Proximal Gradient Algorithm for a Class of Composite Optimization Problems*, *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. S681–S701, 2017.

[12] Guillaume Lauga. *Multilevel proximal methods and application to image restoration*. Mathematics [math]. Ecole normale supérieure de lyon - ENS LYON, 2024. English. ⟨NNT : 2024ENSL0089⟩. ⟨tel-04906328v3⟩

[13] G. Lauga, E. Riccietti, N. Pustelnik, and P. Gonçalves, *Méthodes proximales multi-niveaux pour la restauration d'images*, Nancy, France, Sept. 2022.

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

[15] S. Gratton, M. Mouffe, A. Sartenaer, Ph. L. Toint, and D. Tomanos, *Numerical Experience with a Recursive Trust-Region Method for Multilevel Nonlinear Bound-Constrained Optimization*, Report 08/10, CERFACS and University of Namur, 2008.

[16] A. Beck and M. Teboulle, "Smoothing and First Order Methods: A Unified Framework," *SIAM Journal on Optimization*, vol. 22, pp. 557–580, Jan. 2012.