Harnessing the Structure of some Optimization Problems

Franck Iutzeler

Habilitation à Diriger des Recherches

Harnessing the Structure of some Optimization Problems

Franck Iutzeler

Rapporteurs

Alexandre d'Aspremont	CNRS & École Normale Supérieure
Jérôme Bolte	Université Toulouse Capitole & Toulouse School of Economics
Adrian Lewis	Cornell University

Examinateurs

Jalal Fadili	ENSI Caen
Adeline Leclercq-Samson	Université Grenoble Alpes
Julien Mairal	INRIA Grenoble

Université Grenoble Alpes
Soutenue à Grenoble le 15 Décembre 2021

Preface

The aim of this manuscript is to provide an overview of my research activities during the last five years (or so), *i.e.* 2016–2021. During this period, I was an assistant professor (*Maître de conférences*) at Univ. Grenoble Alpes, in the applied mathematics & computer science "*Jean Kuntzmann*" laboratory.[1]

[1] and still am, as I write these lines.

In the present document, I highlight some of my favorite works in the field of mathematical optimization. Doing so, I focused on two directions: (A) Utilizing the nonsmooth patterns arising in certain data science problems, and (B) Handling multiple asynchronous oracles in distributed minimization. The common thread unifying these two parts lies in the observation that some notion of structure can be observed in these problems and, furthermore, can be used to accelerate the convergence of optimization algorithms.

The organization of the document is straightforward. Chapter 1 provides a general introduction and Chapter 2 recalls the technical tools that will be used throughout the manuscript. Then, Parts A and B contain the core chapters corresponding to the two directions above. Ultimately, Chapter 9 provides some perspectives for future works.

Finally, I adopted a more personal tone during this manuscript compared to my other scientific works. Thus, I included some thoughts and comments on the presented results as well as… disparate quotes and puns. I hope you will enjoy this document in spite of them.

Financial support. This research was partially supported by the French National Research agency (ANR) under JCJC grant *STROLL* (ANR-19-CE23-0008) and by the IDEX Grenoble Alpes under IRS grant *DOLL*.

Remerciements

Merci à mes rapporteurs, Alexandre, Jérôme, et Adrian, pour avoir lu ce manuscrit (je voulais vraiment ne pas dépasser cent pages, je me suis malheureusement laissé dépasser) mais aussi pour leur soutien. Merci aux membres du jury, Jalal, Julien, et Adeline (désolé de te l'avoir demandé de manière étrange !). C'est toujours un plaisir d'échanger avec vous.

Au moment de finaliser et de soutenir cette habilitation, j'ai une pensée particulière pour mes directeurs de thèse, Philippe et Walid, et pour Pascal, avec qui tout a commencé. Merci pour votre enseignement et votre support, ça a été un plaisir de découvrir le monde de la recherche avec vous. J'ai ensuite découvert de nouveaux horizons grâce à Romain, Merouane, et Julien. C'est le genre de voyages qui forment la jeunesse.

Puis j'ai eu la chance d'arriver à Grenoble, dans un laboratoire très accueillant, merci à vous ! Le département Stats/Proba/DATA en particulier, vous êtes super ! Merci à Adeline pour l'accueil, à Francky pour m'avoir initié au trail, à tous les coureurs (Jérôme, Jeff, Sylvain, Charles), aux "jeunes" (Vincent, Julien, Clovis, Charles, Clément). Je pense également aux potes rencontrés en confs, voyages, et workshops (Joseph, Aurélien, Ievgen, Guillaume, Nicolas, Samuel, Édouard, et tant d'autres) qui rendent la communauté scientifique agréable.

Scientifiquement, je suis infiniment redevable à Adeline, Anatoli, Laurent, Massih, Panayotis, Roland, et de nombreux autres auprès de qui je m'excuse pour cet oubli. Ce travail n'aurait pas été possible sans les fantastiques étudiants avec qui j'ai eu la chance de travailler : Bikash, Gilles, Konstantin, Mathias, Mitya, Waïss, Yassine, Yu-Guan ; j'ai hâte de découvrir vos futurs travaux. Jérôme (eh non je t'avais pas oublié !), merci pour tout, tes conseils (même si je suis une tête de mule), ta confiance, et pour m'avoir fait découvrir toute les richesses de l'optimisation. Cette habilitation t'es dédiée.

Plus personnellement, je pense en écrivant ces lignes à mes amis. De Besac à Paris puis à Grenoble en passant par la Belgique. Enfin, je réserve ces derniers mots à ma famille. Mes parents tout d'abord qui ont cru en moi, même quand ils ne voyaient pas bien où j'allais. Moka, qui a animé les réunions Zoom du confinement. Et Cassandre, qui m'apporte tellement chaque jour.

Abstract

Mathematical optimization is becoming more and more important in data science. This is partly due to the increasing difficulty of learning tasks but also to the particular structure of the associated minimization problems which makes them often tractable, sometimes distributable, but always interesting. This is the central thread of this habilitation.

In the first part, I study the mathematical characterization of the underlying structure of the solutions of regularized problems (*e.g.* when a sparsity prior is added to the problem) as well as in the algorithmic exploitation of this phenomenon. The second part of this work deals with the resolution of minimization problems by several machines coordinated asynchronously by a central entity; this type of computation is again made possible by the particular structure of data science problems. Finally, some perspectives conclude this work.

Résumé

L'optimisation mathématique tient une place de plus en plus importante en science des données. Ceci est dû en partie à la difficulté croissante des taches d'apprentissage mais aussi à la structure particulière des problèmes de minimisation associés qui les rend souvent tractables, parfois distribuables, mais toujours intéressants. C'est le thème central de cette habilitation.

Dans un premier temps, je m'intéresse à la caractérisation mathématique de la structure sous-jacente des solutions de problèmes régularisés (par exemple, lorsqu'un a priori de parcimonie est ajouté au problème) ainsi qu'a l'exploitation algorithmique de ce phénomène. La seconde partie de ce document traite de la résolution de problèmes de minimisation par plusieurs machines coordonnées de manière asynchrone par une entité centrale; ce type de calculs est une nouvelle fois rendu possible par la structure particulière des problèmes en science des données. Finalement, quelques perspectives viennent conclure ce travail.

Contents

1 Introduction

Musée des Beaux-Arts et d'Archéologie de Besançon – *Mosaic of Neptune (IInd century)*

Mathematical optimization can be very broadly defined as the "mathematics of decision making" or the "mathematics of doing better".[2] More precisely, an optimization problem consists in selecting a good element from some set of possibilities with respect to some criterion. Usually, this criterion is the minimization of some function that encodes the badness of the possible points while the searching space is a subset of an Euclidean or Hilbert space. The field of mathematical optimization then consists in analyzing and numerically solving such problems.

[2] as per the words of Jérôme Malick and Jean-Baptiste Hiriart-Urruty respectively.

1.1 Context & Motivation

Optimization plays a central role in many disciplines such as signal processing, machine learning, statistics, game theory, economics, control, computer science, etc. Among this very diverse landscape, we will focus in this part on some specific forms of optimization problems that arise in "data science at large" (*i.e.* some convex hull of signal processing, machine learning, and statistics). For these problems, we will show that their specific structure can be used in order to produce a finer analysis as well as better numerical solutions.

1.1.1 Optimization problems in data science...

A common problem in data science is, given some *dataset* $\mathcal{S}_m$ consisting of m points $(\mathrm{a}_i, b_i) \in \mathcal{A} \times \mathcal{B}$ (often supposed i.i.d. from some distribution $\mathcal{D}$), to find a *prediction function* $P : \mathcal{A} \to \mathcal{B}$ that minimizes some *risk* function R measuring the adequation of P to the data distribution. This general problem encompasses standard tasks such as regression, classification, or clustering (see *e.g.* the textbook (Shalev-Shwartz and Ben-David, 2014)).

In practice, this problem has to be approximated to be computationally tractable. Indeed, the set of all possible prediction functions is hardly tractable and the risk function depends on a potentially unknown data distribution. Thus, we restrict ourselves to *parametrized prediction functions* $P_x : \mathcal{A} \to \mathcal{B}$ where $x \in \Theta \subset \mathbb{R}^n$. Typically, the prediction may be taken linear: $P_x(\mathrm{a}) = \Phi(\mathrm{a})^\top x$ where Φ is a mapping from $\mathcal{A} \to \mathbb{R}^n$ and Θ is some compact set so that the prediction does not overfit the data. The problem can then be reformulated as finding a parameter that minimizes the *empirical risk*:

$$\min_{x \in \Theta} \hat{R}(P_x) = \frac{1}{m} \sum_{i=1}^{m} \ell(b_i, P_x(\mathrm{a}_i)) \tag{ERM}$$

where the *loss* $\ell : \mathcal{B} \times \mathcal{B} \to \mathbb{R}_+$ measures the difference between a true and a predicted output (*e.g.* $\ell(b, b') = \frac{1}{2}\|b - b'\|^2$, $\mathbb{1}_{bb'<0}$, or $\log(1 + \exp(-bb'))$). These loss functions often depend either on i) the statistical modelling of the problem through the (log) likelihood (squared 2-norm for linear regression of points corrupted by a Gaussian noise, 1-norm for Laplace noise); or ii) directly from applications (0/1 for "correct"/"incorrect" predictions in classification, eventually convexified/smoothed to improve stability and tractability).

The empirical risk minimization problem (ERM) thus enables us to obtain a good/optimal parameter $x^\star \in \Theta$ and thus a good prediction function $P_{x^\star}$ for our task. This modelling step enabled us to replace a minimization of an implicit objective over a set of functions to the minimization of a real valued function.

1.1.2 ... and their structure

Let us take a closer look at the form of empirical risk minimization problems.

Smooth objective First of all, the objective of the (ERM) problem is often differentiable. For example, this is the case if the prediction function is linear (or at least differentiable) in the parameter and the loss is taken differentiable. These are common modelling choices. However, this may not always be true. The prediction may be nonsmooth when it relies on some specific architecture, for instance in neural networks. In terms of loss, smooth surrogates are often used when the natural one would be nonsmooth, as in classification. A great advantage of this smoothness is the possibility to rely on gradient-based methods which offer good performance in practice and theory in many situations, especially when the dimensions of the problem are large.

Sum form The independence of the data points and the objective of minimizing the average error encoded by the empirical risk naturally result in a sum over the examples. An advantage of this form is that gradients over different parts of the dataset can be computed separately and then recombined. This structure is for instance explicitly used in the variance-reduced stochastic gradient methods (*e.g.* SAGA, SVRG, MISO, etc.) which are state-of-the-art solvers for a variety of machine learning problems.

Nonsmooth contraints/regularization With the ever-growing collection of data, the size of learning problems has significantly increased, both in terms of number of examples, m, as well as in size of optimized parameter, n. While the increase in number of examples is generally beneficial for the general conditioning of the problem, the increase of the parameter space often makes the problem ill-conditioned and reduces both the interpretability and stability of the model. In order to overcome this issue, a generally admitted solution is to introduce a *prior* on the *structure* of the model x (*e.g.* low norm, low rank, sparsity) and encode it in the constraint set Θ.

However, carefully choosing the constraint set may be difficult since one may not want to enforce a prior at all cost, regardless of the model performance of the implied prediction. A possible remedy is to drop the constraint set and *regularize* the (ERM) in order to promote the prior structure. This consists in adding to the objective a nonsmooth function Ω enforcing the sought structure. Indeed, nonsmoothness of functions *traps* optimal solutions in low-dimensional manifolds: small perturbations in the risk around these points would not break down optimal structure as illustrated by Fig. 1.1. Mathematically, this is due to subdifferentials being set-valued and normal to the associated structure. For instance, the subdifferential of the ℓ_1-norm at a point x is set-valued when x lies on the axes and normal to the set of vectors with the same sparsity pattern as x. This means that by adding a ℓ_1-norm to the (ERM), the minima that are close to an axis will be shifted exactly to the axis, the ℓ_1-norm thus promotes sparsity.

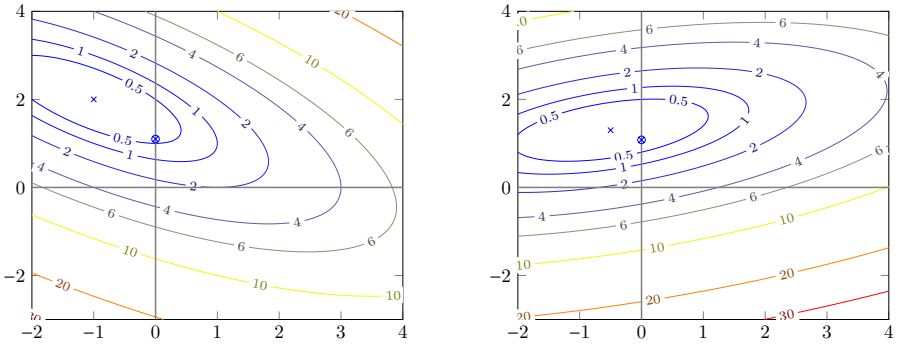


Figure 1.1: Illustration of the stability of optimal solutions of the lasso in $\mathbb{R}^2$. We plot the level sets of $\|Ax - b\|_2^2$ for two problems of (lasso) with different but close design matrices A. We see that while the solutions of the (unregularized) least-squares problem (marked by a $\times$) are different, the solutions of the lasso (marked by a circled cross), although different, lie on the same axis, corresponding to the nonsmoothness loci of the ℓ_1-norm.

Structure of regularized empirical risk minimization At this point, a regularized version of (ERM) can be formulated as

$$\min_{x \in \mathbb{R}^n} \underbrace{\frac{1}{m} \sum_{i=1}^{m} \overbrace{\ell(b_i, P_x(\mathrm{a}_i))}^{=:f^i(x)\ \text{sum form}}}_{\substack{=:f(x) \\ \text{minimizes the risk } \hat{R}(P_x)}} + \underbrace{\lambda\, \Omega(x)}_{\substack{=:g(x) \\ \text{enforces structure}}} \qquad \text{(Regularized ERM)}$$

for some parameter $\lambda > 0$ that controls the balance between fit and structure.

Example 1.1. To make it more concrete, let us consider the popular example of ℓ_1-regularized least-squares problems (often called lasso (Tibshirani, 1996)): take a linear prediction function, a quadratic loss, and the ℓ_1-norm as a regularizer, the regularized empirical risk minimization problem becomes

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2}\|Ax - b\|_2^2 \ + \ \lambda \ \|x\|_1 \ . \tag{lasso}$$

Optimal solutions are then sparser than the ones of the original least-squares, and their sparsity pattern is stable under small perturbations; see Fig. 1.1. ◀

1.2 Contributions presented in this manuscript

Motivated by the problems mentioned above, the central thread of this habilitation will be:[3]

How to harness the | *nonsmooth* / *sum* *structure in optimization problems?*

[3] These works correspond to the core of my research since 2016. My other contributions are summarized in Appendix A and a full publication list can be found in Appendix B.

The center of the manuscript is divided in two parts, corresponding to the two types of structure presented above:

Part A focuses the nonsmooth structure revealed by proximal methods. We start from the observation that in most cases of interest, the proximity operator of the nonsmooth function is explicit as well as its non-differentiability points. This means that under some standard conditions, proximal methods will reach this structure, which is exploitable both numerically and as side-information on the problem. Then, we observe that not all proximal methods are equal with respect structure and provide several ideas to preserve and further numerically exploit this valuable knowledge.

Part B is dedicated to problems with sum structure. Motivated by distributed optimization, we place ourselves in the case where a coordinator can communicate with several workers, each having access to one term of the sum. We focus on the problem of asynchronicity where the workers perform their computation independently and communicate individually with the coordinator as soon as their update is ready, without synchronization. In this setting, we propose efficient methods that explicitly handle delayed information without relying on any information about the computing system such as response times.

Finally, we give in Chapter 9 some perspectives for future research.

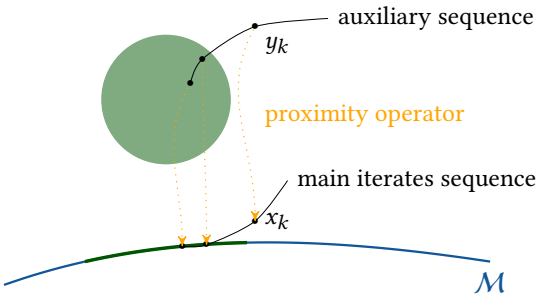
1.2.1 *Part A– Structure Identification in Data Science*

In data science problems, the nonsmoothness often comes from a chosen regularization that enforces some structure. To tackle this specific structure, we formalize three "requirements":

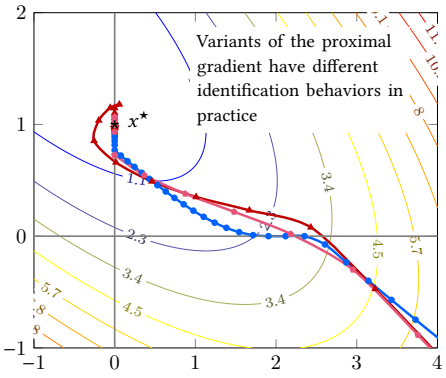
- the proximity operator of the regularization function is computable exactly;
- the structure enforced by the regularization can be described as a collection of Riemannian manifolds;
- as a by-product of the proximity operator computation, the membership to the structure manifolds is also obtained.

These properties are verified by many regularizations/structure couples in practice, the most well-known of which are the ℓ_1-norm/sparsity and the nuclear norm/low rank. The interest of these requirements is that they put us in a favorable case where structure identification and exploitation can be explicitly formulated and put in action.

Chapter 3 draws the mathematical foundations of structure identification for proximal methods. Its goal is to give a simple and intuitive analysis of the process that draws the iterates of an algorithm to a structure-encoding manifold thanks to the use of proximal operations. This chapter contains mostly (variations of) known results; we tried to present them in the most direct and useful way considering the global objective of this part. We start by giving simple conditions for identifying the structure at the limit in a finite number of iterations (either partially or exactly) as well as examples for popular regularizations in data science. However, in order to fully benefit from identification, we also need to control the smoothness of the objective when restricted to the identified structure. This naturally calls for the notion of partial smoothness. After exploring the link between partial smoothness and proximal identification as formulated before, we also show how this property also brings smoothness to the proximity operator.



Chapter 4 focuses on variants of the proximal gradient algorithm. This method is the natural baseline when minimizing the sum of a smooth and a nonsmooth function commonly arising in data science. It is also the starting point for many stochastic and distributed algorithms. The goal of this chapter is to give a practical viewpoint on identification for the main variants of the proximal gradient, focusing on accelerated and monotonous methods. Using the results of the previous chapter, we show that all these methods appear equivalent with respect to finite-time identification. However, numerical experiments reveal striking differences: the fastest methods sometimes have trouble identifying the final structure as they are carried away by their inertia. To combine speed and identification, we propose an inertial proximal gradient method which cancels the inertial acceleration as soon as the current structure is in jeopardy. This kind of methods benefits from the same optimal rate as the accelerated proximal gradient while being much more stable in terms of structure identification as illustrated numerically.

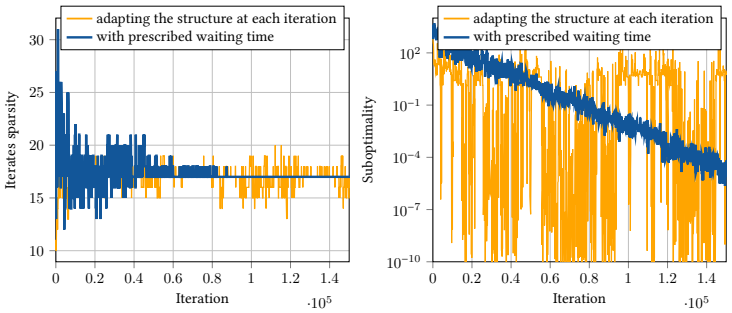


Now that we have both a theoretical and practical intuition about structure identification, a natural question is:

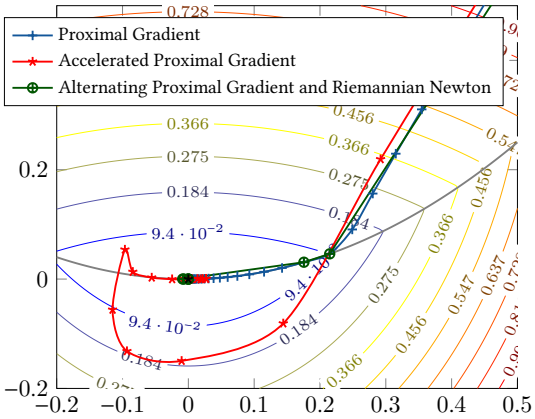
How can we harness the identified structure?

Taking advantage of some given structure can be quite direct by restricting our search space. However, even though the iterates' structure can be observed along the way (as per our "requirements"), we do not know if some structure is final or not. Adaptivity is thus key in our context. Nevertheless, if a manifold is currently identified (*e.g.* some coordinates are null), it is reasonable to postulate that it may be stable at least for some time (*e.g.* these coordinates remain null). Thus, favoring the current structure can provide a numerical boost (*e.g.* updating preferentially the non-null coordinates). Our objective is then to provide methods that adaptively exploit the iterates' structure without putting the theoretical convergence guarantees at risk.

Chapter 5 investigates how identification can be used to update the sampling probabilities in coordinate descent methods. For this chapter, we restrict ourselves to the identification of linear subspaces. We first show how the usual proximal coordinate descent algorithm can be extended to randomly sample along our linear subspaces of interest and investigate its convergence and identification properties. Then, we observe theoretically and in practice that constantly adapting the sampling probabilities to the identified structure leads to a chaotic non-converging algorithm. To overcome this erratic behavior, we devise an adaptive sampling strategy that introduces a waiting time between two consecutive adaptations depending on the amount of change. This restores the convergence and provides a numerical gain in practice compared to non-adaptive methods.



Chapter 6 puts into practice a finer approach to structure exploitation. Previously, we used the fact that the output of a proximity operator is structured. In this chapter, we additionally rely on the notion of partial smoothness which makes the full objective is smooth along the identified manifold. Hence, in order to fully exploit this situation, we turn to Riemannian second-order methods. We show that alternating proximal gradient steps and Riemannian (truncated) Newton steps leads to a structure adaptive methods with super-linear convergence. In practice, we observe a nice numerical boost as soon as a low-dimensional structure is reached.



1.2.2 *Part B – Distributed Structure & Asynchrony*

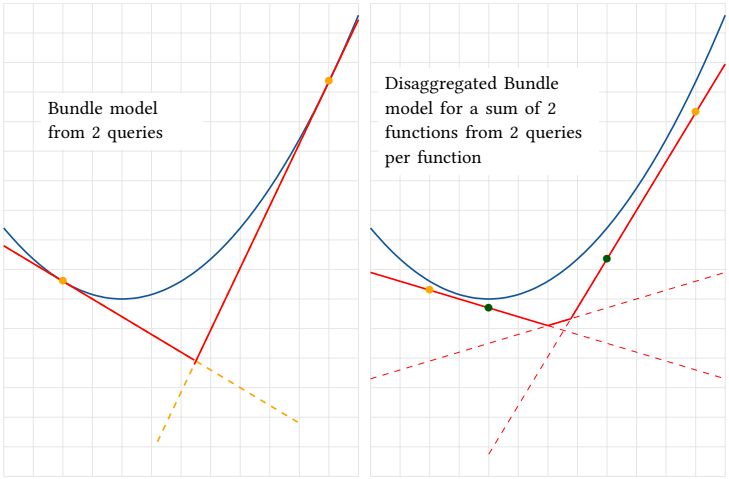
In many optimization problems involving data, the objective features a sum over all the available data. It is the case for empirical risk minimization as mentioned above but also for Lagrangian relaxations for instance. In addition, in modern computing architectures, the data is often scattered over several machines. This means that each machine only has access to part of the data and can produce oracles (*e.g.* gradients) associated with its local batch. When the number of machines grows, the communication between them may become a bottleneck since synchronization implies handling simultaneous access to the medium and waiting for lagging agents. Our key concern is thus:

How to efficiently harness asynchronous oracles?

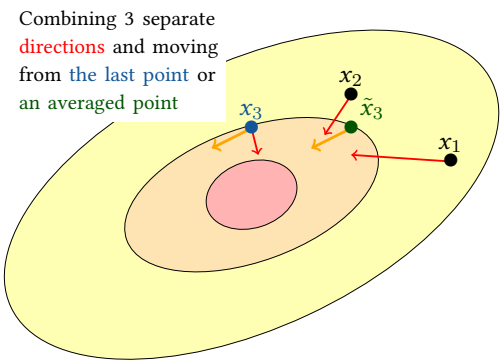
An important point to be resilient to the heterogeneous that appear in practice is how to incorporate information at outdated points. Instead of treating it as a delay-induced noise, our viewpoint is to explicitly handle the older points in the algorithm. As a consequence, we are able to propose methods whose parameters or convergence does not depend on any kind of bound or structure on the delays. The chapters of this part respectively deal with the case where the worker produce subgradient oracles (*i.e.* the nonsmooth case) and gradient oracles.

Chapter 7 deals with the minimization of a sum of nonsmooth functions, each associated with a worker able to produce a subgradient oracles. These kind of problems are typically solved by bundle methods which construct iteratively piecewise-linear lower-approximations of the global function based on (total) subgradient and functional value information. We first show that such approximations can still be obtained from individual information even if the points of query are different (which means that the total functional value or subgradient is not computed at these points so this model

does not "touch" the function graph in general). Thanks to this disaggregated model, we are able to provide two asynchronous bundle methods for which converge can be guaranteed for any kind of delays. The difference between the two aforementioned methods lies in the estimation of an upper-bound on the optimal value: we show that querying all worker at the same point from time to time is highly beneficial in theory and in practice.



Chapter 8 consider the setting of Regularized ERM as a sum of smooth functions, each assigned to one worker, plus a shared nonsmooth function. We present a variant of the proximal gradient that is able to deal with any kind of delays in the response of the asynchronous workers. Beyond the flexibility with respect to the computing setup, we pay a special attention to reducing the communications between the workers and the coordinator. For instance, we allow the workers to perform several gradient steps before communicating their result. Also, when the nonsmooth function is sparsity-inducing, we show how to use the identification properties of the method to sparsify all communications and attain a better performance in terms of communications which is observable both in theory and in practice.



1.3 Reading guide

We end this introduction chapter by a reading guide to help the reader navigate through this document gathering several years of work and around ten papers. The main chapters of the manuscript are divided into two parts (corresponding to the two topics presented above), each featuring a specific introduction to its themes and literature. Inside these parts, the content of the articles are sometimes split in order to factorize the common elements or ideas and to insist on their respective algorithmic or technical contributions.[4] For clarity, each chapter features a short abstract pointing to the associated contributions.

[4] This is also the occasion for additional numerical and graphical illustrations.

In Chapter 2, we review the mathematical tools that will be central to our further results. They are mostly recalls of standard variational analysis results with a special focus on nonsmoothness (subgradients, proximity operators) and Riemannian manifolds (retractions, functional analysis) that enables this manuscript to be almost self-contained. Most readers may probably skip this part or come back to specific results when they are called later in the manuscript.

♦ Part A consist in four chapters regrouping my contributions around proximal identification, which corresponds to the following papers:

A-i F. Iutzeler, J. Malick: Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications, Set-Valued and Variational Analysis, vol. 28, no. 4, pp. 661–678, 2020.

A-ii G. Bareilles, F. Iutzeler: On the Interplay between Acceleration and Identification for the Proximal Gradient algorithm, Computational Optimization and Applications, vol. 77, no. 2, pp. 351–378, 2020.

A-iii F. Iutzeler, J. Malick : On the Proximal Gradient Algorithm with Alternated Inertia , Journal of Optimization Theory and Applications, vol. 176, no. 3, pp. 688-710, 2018.

A-iv D. Grishchenko, F. Iutzeler, J. Malick: Proximal Gradient methods with Adaptive Subspace Sampling, Mathematics of Operations Research, to appear, 2021.

A-v G. Bareilles, F. Iutzeler, J. Malick: Newton acceleration on manifolds identified by proximal-gradient methods, preprint 2021.

Chapter 3 is a tutorial/review chapter on identification for proximal methods, borrowing its viewpoint from *A-i*. The results and intuitions presented there will be the base for the next chapters.

Chapter 4 illustrates the notions introduced in Chapter 3 for the specific case of the proximal gradient and its accelerated variants. In particular, it is the occasion to compare theory and practice, with some remarks and results stemming from *A-ii* and *A-iii*.

Chapter 5 and Chapter 6 correspond to the numerical methods presented in *A-iv* and *A-v* respectively.

♦ Part B has two chapters centered around asynchronous distributed methods which were developed in the following papers:

B-i F. Iutzeler, J. Malick, and W. de Oliveira : Asynchronous level bundle methods, Mathematical Programming, vol. 184, pp. 319-348, 2020.

B-ii K. Mishchenko, F. Iutzeler, J. Malick, M.-R. Amini: A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning, 35-th International Conference on

Machine Learning (ICML), PMLR 80:3584-3592, Stockholm (Sweden), July 2018.

B-iii K. Mishchenko, F. Iutzeler, and J. Malick : A Distributed Flexible Delay-tolerant Proximal Gradient Algorithm , SIAM Journal on Optimization, vol. 30, no. 1, pp. 933-959, 2020.

B-iv D. Grishchenko, F. Iutzeler, J. Malick, and M.-R. Amini: Distributed Learning with Sparse Communications by Identification , SIAM Journal on Mathematics of Data Science, vol. 3, no. 2, pp. 715-735, 2021.

Chapter 7 corresponds to an asynchronous variant of the (level) bundle method presented in *B-i*.

Chapter 8 focuses on the asynchronous proximal gradient method introduced in *B-ii* and studied in *B-iii*. It also discusses the identification-based communication sparsification presented in *B-iv*.

Finally, Chapter 9 presents some perspectives for future research building on the results presented in the manuscript.

2 Preliminaries

Hieronymus Bosch – *The Garden of Earthly Delights (1490-1500)*

The purpose of this chapter is to define the objects and notions at play in the rest of the document. There is no particular contribution in this part and the experienced reader can shamelessly skip part or all of this.

2.1 Variational analysis

In the first page of the renowned book "Variational analysis" by R. Tyrrell Rockafellar and Roger J-B Wets (Rockafellar and Wets, 2009), we are told that "it's convenient for many purposes to consider functions F that are allowed to be extended-real-valued, *i.e.* to take values in $\overline{\mathbb{R}} = [-\infty, +\infty]$ instead of just $\mathbb{R} = (-\infty, +\infty)$", we will thus adopt this convention ourselves.

A fundamental question in variational analysis is the study of the minimum (or equivalently maximum) of functions defined over a Euclidean space $\mathbb{R}^n$. For a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$, we define its *domain* as $\operatorname{dom} F := \{x \in \mathbb{R}^n : F(x) < +\infty\}$, and its *infimum* as

$$\inf F := \inf_{x \in \mathbb{R}^n} F(x) = \inf_{x \in \operatorname{dom} F} F(x).$$

Whenever this infimum is attained, *i.e.* there is some x such that $F(x) = \inf F$, then it is called a *minimum* and is denoted by $\min F$. We further define

$$\operatorname{argmin} F := \{x \in \mathbb{R}^n : F(x) = \inf F\}.$$

Additionally, a function F is *lower semi-continuous* if for any $x \in \mathbb{R}^n$,

$$\liminf_{u \to x} F(x) := \min\{t \in \overline{\mathbb{R}} : \exists u_r \to x \text{ with } F(u_r) \to t\} = F(x).$$

Finally, a function F is said to be *proper* if $F(x) < +\infty$ for at least one $x \in \mathbb{R}^n$ and $F(x) > -\infty$ for all $x \in \mathbb{R}^n$. This means that the domain of a proper function is a nonempty set over which F is finite-valued.

2.1.1 Subgradients

In order to investigate the local behavior of a function with respect to minimization, a first natural step is to consider local affine lower approximations. This *first-order* information is captured by the notion of subgradients. There is a variety of subgradients and several ways to express them, see (Rockafellar and Wets, 2009, Chap. 7,8), (Mordukhovich, 2006, Chap. 1) for general references. We give here only the notions that will be used for our purposes following the terminology and notations of (Rockafellar and Wets, 2009, Chap. 8).

Definition 2.1 (Subgradients)**.** Consider a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \mathbb{R}^n$ at which $F(x)$ is finite:

- the set of *regular subgradients* is defined as

$$\widehat{\partial} F(x) = \{v : F(u) \geq F(x) + \langle v, u - x\rangle + o(\|u - x\|) \text{ for all } u \in \mathbb{R}^n\}, \tag{2.1}$$

- the set of *(general or limiting) subgradients* is defined as

$$\partial F(x) = \left\{\lim_r v_r : v_r \in \widehat{\partial} F(u_r),\ u_r \to x,\ F(u_r) \to F(x)\right\}.$$

If $F(x)$ is infinite, $\widehat{\partial} F(x) = \partial F(x) = \emptyset$.

While the regular subgradient seems simpler and more appealing at first, we will use the general subgradient in all the following, simply referenced under the name subgradient for simplicity. The reason for this is its superior continuity properties as stated in the following lemma.

Lemma 2.2 (Rockafellar and Wets 2009, Th. 8.6, Prop. 8.7)**.** *Consider a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \mathbb{R}^n$ at which $F(x)$ is finite, then the sets of regular subgradients $\widehat{\partial} F(x)$ and general subgradients $\partial F(x)$ are closed. Furthermore, the set of*

general subgradients ∂F is outer semi-continuous at x, ie.

$$\limsup_{u \to x \text{ with } F(u) \to F(x)} \partial F(u) := \{v : \exists u_r \to x, \exists\, v_r \to v \text{ with } v_r \in \partial F(u_r)\} \subset \partial F(x)$$

Note that the regular and limiting subgradients may coincide at some point x, we then say that the function is *(Clarke) regular* at x (Rockafellar and Wets, 2009, Def. 7.25, Cor. 8.11). While less natural in its definition, the outer semi-continuity property of the general subgradient allows us, for example, to deduce that any limit point x of a sequence (x_k) satisfies $0 \in \partial F(x)$ if the distance from $\partial F(x_k)$ to 0 vanishes.

The condition $0 \in \partial F(x)$ is particularly interesting since it is related to local minimas by Fermat's rule (see (Rockafellar and Wets, 2009, Th. 10.1)).

Theorem 2.3 (Fermat's rule)**.** *If a proper function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ has a local minimum at x* (i.e. *if there is a neighborhood $\mathcal{U}$ of x such that $F(x) \leq F(u)$ for all $u \in \mathcal{U}$) then* $0 \in \partial F(x)$.

2.1.2 Gradient and smoothness

If a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ is differentiable at a point x, its *gradient* is the vector of $\mathbb{R}^n$ comprised of its partial derivatives. In line with the regular subgradient notation, it can also be defined as

$$\nabla F(x) = \{v : F(u) = F(x) + \langle v, u - x\rangle + o(\|u - x\|) \text{ for all } u \in \mathbb{R}^n\}\,.$$

From this definition, the following lemma directly follows.

Lemma 2.4. *Consider a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \mathbb{R}^n$ at which F is differentiable, then $\nabla F(x) = \widehat{\partial} F(x) \subset \partial F(x)$. If, in addition, F is continuously differentiable around x, then $\nabla F(x) = \partial F(x)$.*
Furthermore, if $F = f + g$ with f continuously differentiable around x and $g(x)$ finite, then $\partial F(x) = \nabla f(x) + \partial g(x)$.

Remark 2.5. In terms of notations, f will indicate a continuously differentiable (*i.e.* C^1) function, while g will indicate a function that may present non-differentiability points (*i.e.* a *nonsmooth* function). Finally, when smoothness plays no role, the notation F will be used. ◀

Note that there is slight discrepancy of terminology in the literature concerning the notion of smoothness for functions. In (Rockafellar and Wets, 2009), it is used for continuously differentiable functions while in numerical optimization (*e.g.* (Beck, 2017)), it is used for functions with Lipschitz-continuous gradients. We will adopt the latter viewpoint. The reason for this is that it allows us to have a quadratic upper approximation of our function, obtained directly from the fundamental theorem of calculus.

Lemma 2.6. *Consider a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ with a L-Lipschitz continuous gradient,*[5] *then for any $x, u \in \mathbb{R}^n$, one has*

$$|f(x) - f(u) - \langle \nabla f(u), x - u\rangle| \leq \frac{L}{2}\|x - u\|^2$$

[5] *In the following, we will call such a function "L-smooth".*

Such a quadratic approximation is often used for gradients steps, *i.e.* taking

$$x = u - \gamma \nabla f(u)$$

for some $\gamma > 0$. Indeed, in that case, Lemma 2.6 gives use

$$f(x) \leq f(u) - \left(\frac{1}{\gamma} - \frac{L}{2}\right) \|x - u\|^2 \tag{2.2}$$

which provides functional descent whenever $\gamma < 2/L$.

2.1.3 Convexity

In minimization problems, the notion of convexity plays a major role with respect to the localization of minimas.

Definition 2.7. A subset C of $\mathbb{R}^n$ is convex if and only if for any $x, u \in C$, $(1-\alpha)x+\alpha u \in C$ for any $\alpha \in (0,1)$.
A function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ is convex if and only if for any $x, u \in \mathbb{R}^n$, $F((1-\alpha)x + \alpha u) \leq (1-\alpha)F(x) + \alpha F(u)$ for any $\alpha \in (0,1)$.

This class of functions comes with several interesting properties, for instance $\operatorname{dom} F$ and $\operatorname{argmin} F$ are convex if F is convex. Furthermore, every local minimum is a global one. This is again captured by the notion of subgradients.

Lemma 2.8 (Rockafellar and Wets 2009, Prop. 8.12)**.** *Consider a convex proper function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \operatorname{dom} F$. Then,*

$$\partial F(x) = \{v : F(u) \geq F(x) + \langle v, u - x\rangle \text{ for all } u \in \mathbb{R}^n\} = \widehat{\partial} F(x) \neq \emptyset.$$

Thus, F is regular at any point and $0 \in \partial F(x)$ if and only if $x \in \operatorname{argmin} F$.

If, in addition, F is differentiable, convexity can be seen directly as a property on the gradient mapping.

Theorem 2.9 (Bauschke and Combettes 2011, Prop. 17.10)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a proper function with open domain. Suppose that f is differentiable on $\operatorname{dom} f$. Then the following are equivalent:*

i) f is convex;

ii) $f(u) \geq f(x) + \langle \nabla f(x), u - x\rangle$ for all $x, u \in \operatorname{dom} f$;

iii) $\langle \nabla f(x) - \nabla f(u), x - u\rangle \geq 0$ for all $x, u \in \operatorname{dom} f$, i.e. *$\nabla f$ is monotone.*

Furthermore, if f is twice differentiable on $\operatorname{dom} f$, any of the above is equivalent to

iv) $\langle u, \nabla^2 f(x) u\rangle \geq 0$ for all $x, u \in \operatorname{dom} f$, i.e. *$\nabla^2 f$ is positive semi-definite.*

Finally, while convexity provides affine lower bounds, having (convex) quadratic lower-bounds enable to get a better control that may have a great impact on the convergence of optimization methods; this is captured by the notion of strong convexity.

Definition 2.10. For some $\mu > 0$, a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ is μ-strongly convex if and only if $F - \frac{\mu}{2}\|\cdot\|^2$ is convex.

Using the fact that $\tilde{F} := F - \frac{\mu}{2}\|\cdot\|^2$ is convex and verifies $\partial \tilde{F} = \partial F - \mu \cdot$ by Lemma 2.4, we get that for any $x \in \mathbb{R}^n$ and any $v \in \partial F(x)$

$$F(u) \geq F(x) + \langle v, u - x\rangle + \frac{\mu}{2}\|u - x\|^2 \text{ for all } u \in \mathbb{R}^n \tag{2.3}$$

which directly implies that a strongly convex function has at most one minimizer by taking x such that $0 \in \partial F(x)$. The following lemma then adds the existence (see (Bauschke and Combettes, 2011, Chap. 11.4) for a more general take).

Lemma 2.11. *Let $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a strongly convex lower semi-continuous proper function, then F has exactly one minimizer.*

Proof. From (2.3), we get that for all $u \in \mathbb{R}^n$,

$$\begin{aligned} F(u) &\geq F(x) + \frac{\mu}{2}\|x\|^2 - \langle v, x\rangle + \langle v + \mu x, u\rangle + \frac{\mu}{2}\|u\|^2 \\ &\geq F(x) + \frac{\mu}{2}\|x\|^2 - \langle v, x\rangle - \|v + \mu x\|\|u\| + \frac{\mu}{2}\|u\|^2 \end{aligned}$$

hence $F(u)/\|u\| \to +\infty$ when $\|u\| \to +\infty$, *i.e.* F is supercoercive. Thus, this means that for any t, the level set $\{x : F(x) \leq t\}$ is bounded (this is direct by contradiction, see (Bauschke and Combettes, 2011, Prop. 11.11)). This means that since F is proper, we can take t sufficiently large so that the corresponding level set is non-empty and bounded. Finally, since F is lower semi-continuous, applying Weierstrass extreme value theorem (see (Bauschke and Combettes, 2011, Th. 1.28) in our case) to this compact set gives us the existence of a minimal value, which is unique from the quadratic lower bound expressed in (2.3). □

2.2 The proximity operator

A central tool to tackle non-differentiable functions is the *proximity operator* introduced by Jean-Jacques Moreau in (Moreau, 1962, 1965). For a (nonsmooth) function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a step-size $\gamma > 0$, the proximity operator $\mathrm{prox}_{\gamma g}$ is defined as the set-valued mapping

$$\mathrm{prox}_{\gamma g}(y) := \mathrm{argmin}_{u \in \mathbb{R}^n} \underbrace{\left\{ g(u) + \frac{1}{2\gamma}\|u - y\|^2 \right\}}_{:=\rho_y(u)}. \tag{2.4}$$

In the same flavor as for the gradient step, if one takes a proximal step, *i.e.*

$$x = \mathrm{prox}_{\gamma g}(y)$$

for some $\gamma > 0$, the definition directly gives us

$$g(x) \leq g(y) - \frac{1}{2\gamma}\|x - y\|^2 \tag{2.5}$$

which mirrors (2.2) (the descent inequality of a gradient step on a smooth function) but for a potentially nonsmooth function. Actually, this link can be made formal since a proximal step is equivalent to a gradient step on the *Moreau envelope* defined for all $y \in \mathbb{R}^n$ as $e_\gamma g(y) = \inf_{u \in \mathbb{R}^n} \rho_y(u)$ (Moreau, 1965; Yosida, 1988). Another possible viewpoint is to notice that $x = y - \gamma \partial g(x)$ which is sometimes called an implicit gradient step to emphasize its relation with numerical integration methods for ordinary differential equations.

In that respect, the proximity operator provides an alternative to the use of subgradients since they are not able to provide sufficient descent inequalities such as (2.2) and (2.5). However, this comes with the cost of having to solve a minimization subproblem, which in turn raises questions about the existence and uniqueness of the subproblem solutions.

Remark 2.12 (Convex case). When $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a convex lower semi-continuous proper function, everything is made easy since Lemma 2.11 guarantees the existence and uniqueness of the minimizers of $\rho_y(u)$ for any u, which means that $\operatorname{prox}_{\gamma g}(y)$ is well-defined and unique. ◂

Contrary to the convex case, the existence and even more, the unicity in the general case mainly comes from scattered results in the literature. Hence, we try to regroup them here in a self-contained manner.

2.2.1 Existence

Proximity operators exist as soon as the minimization subproblem has a solution. For this, the function g needs to be proper (to have finite values) and lower semi-continuous (so that the infimum over a compact set can be attained). But that is not sufficient, the function g must not decrease to $-\infty$ too fast: if $g(y) = -\exp(y)$, the problem has no solution; if $g(y) = -r\|y\|^2$, the problem may or may not have a solution, depending on the value of r, finally if $g(y) \geq \beta > -\infty$ then the problem always has a solution. This behavior with respect to quadratic functions is captured by the notion of *prox-boundedness* (see (Rockafellar and Wets, 2009, Chap. 1.G) for a detailed treatment).

Definition 2.13. A proper lower semi-continuous function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is r_{pb}-prox-bounded if either of the equivalent conditions hold:

- $e_\gamma g(y) > -\infty$ for some $y \in \mathbb{R}^n$ and all $0 < \gamma \leq 1/r_{pb}$;
- $g + \frac{1}{2\gamma}\|\cdot\|^2$ is bounded from below on $\mathbb{R}^n$ for all $0 < \gamma < 1/r_{pb}$.

In particular, if g is bounded from below on $\mathbb{R}^n$, *i.e.* $\inf g > -\infty$, then g is prox-bounded with threshold $r_{pb} = 0$.

This definition allows us to show the existence of the proximity operator.

Theorem 2.14. *Let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper lower semi-continuous function. If g is r_{pb}-prox-bounded, then for any $\gamma \in (0, 1/r_{pb})$, $\operatorname{prox}_{\gamma g}(y)$ is non-empty and compact for all $y \in \mathbb{R}^n$.*

Proof. Following the same line as (Rockafellar and Wets, 2009, Th. 1.25), we use the prox-boundedness to show that for any y, and any $\gamma \in (0, 1/r_{pb})$, $\rho_y(u) \geq \beta + c\|y - u\|^2$ for any y and some $\beta \in \mathbb{R}$, $c > 0$. The surrogate ρ_y is thus coercive (that is $\rho_y(u) \to +\infty$ whenever $\|u\| \to +\infty$) and thus has bounded level sets (see (Bauschke and Combettes, 2011, Prop. 11.11)). These level sets are non-empty above some value since ρ_y is proper (which is directly implied by g being proper) and closed since ρ_y is lower semi-continuous (Bauschke and Combettes, 2011, Lem. 1.24) (as g is lower semi-continuous, the squared norm is continuous and the positive sum of lower semi-continuous functions is lower semi-continuous (Bauschke and Combettes, 2011, 1.27)). Finally, restricting ρ_y to a non-empty closed level set, the infimum is attained by the extreme value theorem (Bauschke and Combettes, 2011, Th. 1.28) which means that $\operatorname{prox}_{\gamma g}(y)$ is non-empty. The closedness comes from the lower semi-continuity of g and the boundedness from the coercivity of ρ_y. □

From this result, we see that the existence of proximity operators is usually direct for the user. The only point on which one has to be careful is the potential limit on the range of stepsizes whenever g is not lower bounded.

2.2.2 Prox-regularity

Once existence is taken care of, the next natural question is the uniqueness and the stability of proximal point, at least locally. For this, it is natural to express the fact that g locally has a quadratic lower model so that the sensitivity of the minimizers of ρ_y is locally controlled.[6] Otherwise said, one has to add a bit of control in the small o of the regular subgradient inequality (2.1) locally around some point. This is captured by the notion of *prox-regularity* (Rockafellar and Wets, 2009, Chap. 13.F).

[6] Here, opposedly to strong convexity, this is a concave quadratic lower model.

Definition 2.15. A function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is r-prox-regular at $\bar{x}$ for $\bar{v} \in \partial g(\bar{x})$ if g is finite and locally lower semi-continuous at $\bar{x}$ and there exists $\varepsilon > 0$ such that

$$g(u) \geq g(x) + \langle v, u - x\rangle - \frac{r}{2}\|u - x\|^2 \text{ for all } u \in \mathcal{B}(\bar{x}, \varepsilon) \tag{2.6}$$

when $v \in \partial g(x)$, $\|v - \bar{v}\| < \varepsilon$, $\|x - \bar{x}\| < \varepsilon$, and $g(x) < g(\bar{x}) + \varepsilon$.
When the above holds for all $\bar{v} \in \partial g(\bar{x})$, g is said to be r-prox-regular at $\bar{x}$.

The first thing to notice is that it is a local property around $\bar{x}$ but also for close values of the subgradients and the functions. For instance, one can check that the $\mathbb{R} \to \mathbb{R}$ function $\mathrm{ind}_{\{0\}}$ defined as $\mathrm{ind}_{\{0\}}(x) = 1$ for all $x \neq 0$ and $\mathrm{ind}_{\{0\}}(0) = 0$ is prox-regular with any threshold, but x can only be taken equal to $\bar{x}$ in (2.6). We note also that if a function is r-prox-regular then it is r'-prox-regular for all $r' \geq r$.

In addition, taking $x = \bar{x}$ in (2.6), we deduce that if g is r-prox-regular at $\bar{x}$, it is also (Clarke, or subdifferentially) regular[7] at $\bar{x}$.

[7] However, this does not hold if g is r-prox-regular at $\bar{x}$ only for some subgradients.

2.2.3 Unicity and Lipchitz continuity

Now, given some point $y \in \mathbb{R}^n$ and coming back to the definition of the proximity operator,

$$\begin{aligned}
& x \in \mathrm{prox}_{\gamma g}(y) \\
\Leftrightarrow \quad & g(x) + \frac{1}{2\gamma}\|x - y\|^2 \leq g(u) + \frac{1}{2\gamma}\|u - y\|^2 \text{ for all } u \in \mathbb{R}^n && (2.7)\\
\Leftrightarrow \quad & g(u) \geq g(x) + \langle \underbrace{(y - x)/\gamma}_{:= v}, u - x\rangle - \frac{1}{2\gamma}\|u - x\|^2 \text{ for all } u \in \mathbb{R}^n && (2.8)\\
\Leftrightarrow \quad & x \in \mathrm{prox}_{\gamma g}(x + \gamma v) \\
\Rightarrow \quad & v = \frac{y - x}{\gamma} \in \partial g(x) && (2.9)
\end{aligned}$$

where the last implication comes from the generalized Fermat's rule (Rockafellar and Wets, 2009, 10.1) since x is a minimum of ρ_y.[8]

[8] Fermat's rule implies that $0 \in \partial \rho_y(x) = \partial g(x) + 1/\gamma(x - y)$.

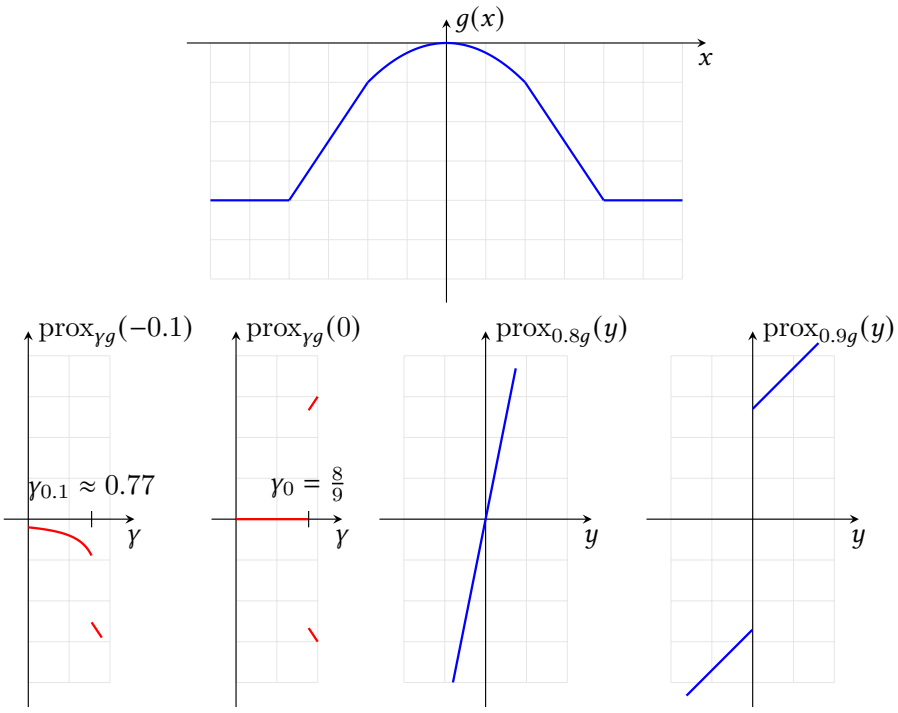
A natural question is thus, given two points $x, y \in \mathbb{R}^n$ such that $(y - x)/\gamma \in \partial g(x)$, do we have $x = \mathrm{prox}_{\gamma g}(y)$ (with unicity)? The notion of prox-regularity seems particularly helpful here. However, looking at (2.6) and (2.8), we see that the former is local while the later is global; this means that prox-regularity ensures only (2.8) around some $\bar{x}$. The inequality thus has to be globalized.

This question has been widely addressed in the literature, stemming mainly from (Poliquin and Rockafellar, 1996, Th. 4.4) (see also (Hare and Sagastizábal, 2009, Th. 4) for a more recent take). However, a less known fact is that the range of proximal stepsizes allowed depends if the points are around a proximal output (*i.e.* a point $\bar{x}$

such that there is some $\bar{y}$ and $\bar{\gamma}$ such that $\bar{x} = \mathrm{prox}_{\bar{\gamma} g}(\bar{y})$). If such a point exists, γ can be taken in $(0, \min(1/r, \bar{\gamma}))$, otherwise it has to be taken sufficiently small. The following example illustrates this difference.

Example 2.16 (Prox-regularity does not necessarily imply single-valued and Lipschitz continuity). We consider the $\mathbb{R} \to \mathbb{R}$ function

$$g(x) = \max\left(-2, \min\left(-\frac{x^2}{2}, 1 - \frac{3|x|}{2}\right)\right) = \begin{cases} -2 & x \leq -2 \\ 1 + \frac{3x}{2} & x \in [-2, -1] \\ -\frac{x^2}{2} & x \in [-1, 1] \\ 1 - \frac{3x}{2} & x \in [1, 2] \\ -2 & x \geq 2 \end{cases}$$



g is $r = 1$-prox-regular at 0 for $0 \in \partial g(0) = \{0\}$ (and $\varepsilon = 1$). Thus, one can expect taking $\gamma \in (0, 1)$. However, we can show that for any $y \in [-1 + \gamma, 1 - \gamma]$, there is a threshold stepsize

$$\gamma_y = \frac{17 - 12|y| - \sqrt{24|y| + 1}}{18} \in \left[\frac{2}{3}, \frac{8}{9}\right]$$

such that for $y \neq 0$

$$\mathrm{prox}_{\gamma g}(y) = \begin{cases} \frac{y}{1-\gamma} & \text{for } \gamma < \gamma_y \\ y + \mathrm{sign}(y)\frac{3}{2}\gamma & \text{for } \gamma > \gamma_y \\ \{\frac{y}{1-\gamma}, y + \mathrm{sign}(y)\frac{3}{2}\gamma\} & \text{for } \gamma = \gamma_y \end{cases}$$

and for $y = 0$

$$\operatorname{prox}_{\gamma g}(0) = \begin{cases} 0 & \text{for } \gamma < \frac{8}{9} \\ \{-\frac{3}{2}\gamma, \frac{3}{2}\gamma\} & \text{for } \gamma > \frac{8}{9} \\ \{-\frac{3}{2}\gamma, 0, \frac{3}{2}\gamma\} & \text{for } \gamma = \frac{8}{9} \end{cases}.$$

This means that when $0 < \gamma < \frac{8}{9}$, one can find a neighborhood of 0 where the proximal operator is Lipschitz continuous: $\{y : |y| < (3(1-\gamma) - \sqrt{1-\gamma})/2\}$. However, this is not the case for $\frac{8}{9} \leq \gamma < 1$ where Lipschitz continuity does not hold and the proximity operator may not be single valued. ◀

Thus, without assuming a neighboring proximal point, the following result can be found in (Rockafellar and Wets, 2009, Prop. 13.37).

Theorem 2.17. *Suppose that $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is r-prox-regular at $\bar{x}$ for $\bar{v} = 0$ and that g is prox-bounded. Then, for all $\gamma > 0$* sufficiently small*, there is a neighborhood of $\bar{x}$ on which $prox_{\gamma g}$ is single-valued and $(1-\gamma r)^{-1}$-Lipschitz-continuous.*

Since we will need Lipschitz continuity and the subdifferential characterization of (2.9) for a wider range of proximal stepsizes, we will require the existence of a close proximal point and the following theorem.[9]

Theorem 2.18. *Consider a function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$, a pair of points $\bar{x}, \bar{y}$ and a stepsize $\bar{\gamma} > 0$ such that*

i) $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$*;*

ii) g is r-prox-regular at $\bar{x}$ for subgradient $\bar{v} := (\bar{y} - \bar{x})/\bar{\gamma} \in \partial g(\bar{x})$.[10]

Then, for any $0 < \gamma < \min(\frac{1}{r}, \bar{\gamma})$ there exists $\varepsilon > 0$ such that when $\|\bar{y} - \bar{x}\| \left(\frac{1}{\gamma} - \frac{1}{\bar{\gamma}}\right) \leq \varepsilon$:

a) for all $x \in \mathcal{B}(\bar{x}, \varepsilon), y \in \mathcal{B}(\bar{y}, \varepsilon), g(x) \leq g(\bar{x}) + \varepsilon$ and $\|(y - x)/\gamma - \bar{v}\| < \varepsilon$,

$$v = \frac{y - x}{\gamma} \in \partial g(x) \Leftrightarrow x = \operatorname{prox}_{\gamma g}(x + \gamma v) \Leftrightarrow x = \operatorname{prox}_{\gamma g}(y);$$

b) $\operatorname{prox}_{\gamma g}$ is locally $(1-\gamma r)^{-1}$-Lipchitz continuous on $\mathcal{B}(\bar{y}, \varepsilon)$.

Note here that the condition $\gamma \in (0, \min(1/r, \bar{\gamma}))$ may be misleading since for the result to hold, the conditions $\|(y - x)/\gamma - \bar{v}\| < \varepsilon$ and $y \in \mathcal{B}(\bar{y}, \varepsilon)$ also have to be fulfilled. This means that the quantity $\|\bar{y} - \bar{x}\| (1/\gamma - 1/\bar{\gamma})$ also has to be small, we chose to put it explicitly in the above result to emphasize this point. This can be done either by taking γ sufficiently close to $\bar{\gamma}$ (which may not be possible since γ has to be smaller than $1/r$); or when $\|\bar{y} - \bar{x}\|$ are sufficiently small, *i.e.* around fixed points of the proximal operator.

Proof. The proof is divided in several parts. First, we show that $v = \frac{y-x}{\gamma} \in \partial g(x) \Rightarrow x = \operatorname{prox}_{\gamma g}(x + \gamma v)$; since the converse is immediate by (2.9), this gives the first equivalence. Then, we show how to convert y into $x + \gamma v$ to get the second equivalence. Finally, we show the local Lipschitz continuity.

Step 1: $v = \frac{y-x}{\gamma} \in \partial g(x) \Rightarrow x = \operatorname{prox}_{\gamma g}(x + \gamma v)$

Let γ denote a scalar in $(0, \min(1/r, \bar{\gamma}))$, and $x, v \in \partial g(x)$ denote a pair close to $\bar{x}, \bar{v}$ in the sense that $\|x - \bar{x}\| \leq \varepsilon_x$, $g(x) \leq g(\bar{x}) + \varepsilon_g$, and $\|v - \bar{v}\| \leq \varepsilon_v$. We want to show that, for small enough values of $\varepsilon_x, \varepsilon_g, \varepsilon_v$, we have $x = \operatorname{prox}_{\gamma g}(x + \gamma v)$, that is

$$g(x) + \langle v, u - x \rangle - \frac{1}{2\gamma}\|u - x\|^2 < g(u) \qquad \text{for all } u \in \mathbb{R}^n \tag{2.10}$$

[9] While not new per se, this theorem extends several results of the literature. Additionally, we present an original self-contained proof.

[10] *Note that $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$ implies that $(\bar{y} - \bar{x})/\bar{\gamma} \in \partial g(\bar{x})$, see* (2.9).

where the inequality is strict to indicate that x is the unique output of $\operatorname{prox}_{\gamma g}(x+\gamma v)$.

Note that if u lies in a ball of radius ε centered at $\bar{x}$, (2.10) holds since $-1/\gamma < -r$ and g is r-prox-regular at $\bar{x}$ for subgradient $\bar{v}$ (see (2.6)). We thus focus on the case $u \notin \mathcal{B}(\bar{x}, \varepsilon)$.

First,

$$\begin{aligned}\langle v, u-x\rangle &= \langle v-\bar{v}, u-x\rangle + \langle \bar{v}, u-x\rangle \\ &\leq \varepsilon_v\|u-x\| + \langle \bar{v}, u-\bar{x}\rangle + \langle \bar{v}, \bar{x}-x\rangle \\ &\leq \varepsilon_v\varepsilon_x + \varepsilon_v\|u-\bar{x}\| + \langle \bar{v}, u-\bar{x}\rangle + \|\bar{v}\|\varepsilon_x.\end{aligned}$$

Using this inequality along with $g(x) < g(\bar{x}) + \varepsilon_g$ yields:

$$\begin{aligned}&g(x) + \langle v, u-x\rangle - \frac{1}{2\gamma}\|u-x\|^2 \\ &\leq g(\bar{x}) + \varepsilon_g + \varepsilon_v\varepsilon_x + \varepsilon_v\|u-\bar{x}\| + \langle \bar{v}, u-\bar{x}\rangle + \|\bar{v}\|\varepsilon_x - \frac{1}{2\gamma}\|u-x\|^2\end{aligned}$$

The global optimality of $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$ writes $g(\bar{x}) + \langle \bar{v}, u-\bar{x}\rangle - \frac{1}{2\bar{\gamma}}\|u-\bar{x}\|^2 \leq g(u)$ for all $u \in \mathbb{R}^n$ (see (2.8)). Thus, we have

$$g(x)+\langle v, u-x\rangle-\frac{1}{2\gamma}\|u-x\|^2 \leq g(u)+\varepsilon_g+\varepsilon_v\varepsilon_x+\varepsilon_v\|u-\bar{x}\|+\|\bar{v}\|\varepsilon_x-\frac{1}{2\gamma}\|u-x\|^2+\frac{1}{2\bar{\gamma}}\|u-\bar{x}\|^2,$$

and conclude using that $-\|u-x\|^2 = -\|u-\bar{x}\|^2 - \|\bar{x}-x\|^2 - 2\langle u-\bar{x}, \bar{x}-x\rangle \leq -\|u-\bar{x}\|^2 + 2\|u-\bar{x}\|\varepsilon_x$ to get

$$\begin{aligned}& g(x) + \langle v, u-x\rangle - \frac{1}{2\gamma}\|u-x\|^2 \\ \leq\ & g(u) + \varepsilon_g + \varepsilon_v\varepsilon_x + \varepsilon_v\|u-\bar{x}\| + \|\bar{v}\|\varepsilon_x - \frac{1}{2\gamma}\|u-\bar{x}\|^2 + \frac{1}{\gamma}\|u-\bar{x}\|\varepsilon_x + \frac{1}{2\bar{\gamma}}\|u-\bar{x}\|^2 \\ =\ & g(u) + \underbrace{\left(\frac{1}{2\bar{\gamma}} - \frac{1}{2\gamma}\right)\|u-\bar{x}\|^2 + \left(\frac{\varepsilon_x}{\gamma} + \varepsilon_v\right)\|u-\bar{x}\| + \varepsilon_g + \varepsilon_v\varepsilon_x + \|\bar{v}\|\varepsilon_x}_{:= q_{\varepsilon_x,\varepsilon_g,\varepsilon_v}(\|u-\bar{x}\|)}\end{aligned}$$

As a second step, we show that for small enough values $\varepsilon_x, \varepsilon_g, \varepsilon_v$, the value of $q_{\varepsilon_x,\varepsilon_g,\varepsilon_v}$ is strictly negative over $[\varepsilon, +\infty[$ (that is if $\|u-\bar{x}\| \geq \varepsilon$ which corresponds to our case $u \notin \mathcal{B}(\bar{x}, \varepsilon)$). This is equivalent to restricting the size of the neighborhood of $\bar{x}, \bar{v}$ on which x and v can be chosen. First, note that q is strictly concave since $\frac{1}{2\bar{\gamma}} - \frac{1}{2\gamma} < 0$, or equivalently $0 \leq \gamma < \bar{\gamma}$. Negativity of q over $[\varepsilon, +\infty)$ is thus ensured if the maximum of q is attained below ε, and $q(\varepsilon) < 0$, which writes

$$\begin{cases} (\frac{\varepsilon_x}{\gamma} + \varepsilon_v)(\gamma^{-1} - \bar{\gamma}^{-1})^{-1} \leq \varepsilon \\ \left(\frac{1}{2\bar{\gamma}} - \frac{1}{2\gamma}\right)\varepsilon^2 + \left(\frac{\varepsilon_x}{\gamma} + \varepsilon_v\right)\varepsilon + \varepsilon_g + \varepsilon_v\varepsilon_x + \|\bar{v}\|\varepsilon_x < 0 \end{cases}$$

As we know that $\frac{1}{2\bar{\gamma}} - \frac{1}{2\gamma} < 0$, these equations hold for small enough values of $\varepsilon_x, \varepsilon_g, \varepsilon_v$. We have thus proved strict negativity of $q_{\varepsilon_x,\varepsilon_g,\varepsilon_v}$ which implies that, for all $u \in \mathbb{R}^n$:

$$g(x) + \langle v, u-x\rangle - \frac{1}{2\gamma}\|u-x\|^2 < g(u),$$

or equivalently, $x = \operatorname{prox}_{\gamma g}(x + \gamma v)$ and in particular $\operatorname{prox}_{\gamma g}(x + \gamma v)$ is unique.

Step 2: Correspondance between $x + \gamma v$ and y

Let $x \in \operatorname{prox}_{\gamma g}(y)$. We have by (2.9) that $v = \frac{y-x}{\gamma} \in \partial g(x)$. Using (2.7) at $\bar{x}$, and doing the same for $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$ at x, we get:

$$\begin{aligned} g(x) + \frac{1}{2\gamma}\|x - y\|^2 &\leq g(\bar{x}) + \frac{1}{2\gamma}\|\bar{x} - y\|^2 \\ g(\bar{x}) + \frac{1}{2\bar{\gamma}}\|\bar{x} - \bar{y}\|^2 &\leq g(x) + \frac{1}{2\bar{\gamma}}\|x - \bar{y}\|^2 \end{aligned} \tag{2.11}$$

and adding these two inequalities, we get

$$\begin{aligned} & \frac{\bar{\gamma} - \gamma}{\bar{\gamma}\gamma}\|x - \bar{x}\|^2 \leq 2\langle x - \bar{x}, (\bar{y} - \bar{x})\left(\frac{1}{\gamma} - \frac{1}{\bar{\gamma}}\right)\rangle + 2\langle x - \bar{x}, (y - \bar{y})\frac{1}{\gamma}\rangle \\ \Rightarrow \quad & \|x - \bar{x}\| \leq 2\frac{\bar{\gamma}\gamma}{\bar{\gamma} - \gamma}\left(\frac{1}{\gamma} - \frac{1}{\bar{\gamma}}\right)\|\bar{y} - \bar{x}\| + \frac{2}{\gamma}\|y - \bar{y}\| \end{aligned}$$

and thus $\|x - \bar{x}\|$ can be made as small as one want by controlling ε for y close enough to $\bar{y}$. Furthermore, this implies that v is close to $\bar{v}$ and that $g(x)$ is close to $g(\bar{x})$ by (2.11). Thus, the reasonning of Step 1 can be readily carried with x, v. Note finally that we did not need any assumption on x or v.

Step 3: Local Lipchitz continuity

Finally, let x, u, y, z be four points such that $x = \operatorname{prox}_{\gamma g}(y)$ and $u = \operatorname{prox}_{\gamma g}(z)$. When y, z are near $\bar{y}$, this implies that x and u are close to $\bar{x}$ and globally all assumptions for prox-regularity are satified by the arguments of Step 2. Then, prox-regularity at point $\bar{x}$ allows to write for points x, u and subgradients $v_x = \frac{y-x}{\gamma} \in \partial g(x)$, $v_u = \frac{z-u}{\gamma} \in \partial g(u)$:

$$\begin{aligned} g(x) &\geq g(u) + \langle v_u, x - u\rangle - \frac{r}{2}\|x - u\|^2 \\ g(u) &\geq g(x) + \langle v_x, u - x\rangle - \frac{r}{2}\|u - x\|^2 \end{aligned}$$

Summing yields $\langle v_x - v_u, x - u\rangle \geq -r\|x - u\|^2$ and replacing the chosen subgradients by their expressions gives $\langle (y - x) - (z - u), x - u\rangle \geq -r\gamma\|x - u\|^2$, which rewrites $\langle y - z, x - u\rangle \geq (1 - r\gamma)\|x - u\|^2$ and applying Cauchy-Schwarz leads to

$$(1 - \gamma r)\|x - u\| \leq \|y - z\|.$$

Since $\gamma < 1/r$, this implies that $\|x - u\| \leq (1 - \gamma r)^{-1}\|y - z\|$, thus $\operatorname{prox}_{\gamma g}$ is $(1 - \gamma r)^{-1}$-Lipschitz continuous around $\bar{y}$. □

Remark 2.19 (Proof using directly (Poliquin and Rockafellar, 1996)). In the spirit of (Hare and Sagastizábal, 2009, Th. 4), the above result can also be proved by directly using (Poliquin and Rockafellar, 1996) but the proof bears less intuition as the one provided above. First, one can easily check that prox-regularity of g at $\bar{x}$ for subgradient $\bar{v}$ is equivalent to prox-regularity of function $\tilde{g}$ around 0 for subgradient 0, with $\tilde{g} = g(\cdot + \bar{x}) - \langle \bar{v}, \cdot\rangle - g(\bar{x})$ and a change of variable $\tilde{x} = x - \bar{x}$. Similarly, $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$ is characterized by its global optimality condition (2.8) which rewrites with our notation

$$\tilde{g}(\tilde{x}) > -\frac{1}{2\bar{\gamma}}\|\tilde{x}\|^2 \qquad \text{for all } \tilde{x} \neq 0.$$

We may thus apply Theorem 4.4 from (Poliquin and Rockafellar, 1996) to get the claimed result on $\tilde{g}$, which transfers back to g as our change of function and variable is bijective. We thus obtain that for $\gamma \in (0, \min(1/r, \bar{\gamma}))$, on a neighborhood $\mathcal{N}_{\bar{y}}$ of $\bar{y}$, $\mathrm{prox}_{\gamma g}$ is single-valued, $(1-\gamma r)^{-1}$-Lipschitz continuous and $\mathrm{prox}_{\gamma g}(y) = [I + \gamma T]^{-1}(y)$, where T denotes the g-attentive ε-localization of $\partial g(\bar{x})$ (see (Rockafellar and Wets, 2009, Chap. 13.F)). Taking y near $\bar{y}$ and x near $\bar{x}$ such that $\|x - \bar{x}\| < \varepsilon$, $|g(x) - g(\bar{x})| < \varepsilon$ and $\|(y-x)/\gamma - \bar{v}\| < \varepsilon$ allows to identify the localization of $\partial g(x)$ with $\partial g(x)$, so that

$$\frac{y-x}{\gamma} \in \partial g(x) \Leftrightarrow \frac{y-x}{\gamma} \in T(x) \Leftrightarrow (I+\gamma T)(x) = y \Leftrightarrow x = \mathrm{prox}_{\gamma g}(y).$$

Note that the proof of (Poliquin and Rockafellar, 1996, Th. 4.4) has a minor error relative to the Lipschitz constant computation, we report here the corrected value. ◂

Remark 2.20. Theorem 2.18 also implies the following inequality for two points y, z close to $\bar{y}$, with $x = \mathrm{prox}_{\gamma g}(y)$, $u = \mathrm{prox}_{\gamma g}(z)$:

$$(1 - 2r\gamma)\|x-u\|^2 \leq \|y-z\|^2 - \|(x-y)-(u-z)\|^2 \tag{2.12}$$

which generalizes the firm non-expansiveness characterization of the proximity operator of a convex function (Bauschke and Combettes, 2011, Prop. 12.27). ◂

2.2.4 Strong local minimizers

An additional property of prox-regular functions is that their critical points are *strong minimizers* of their surrogate ρ. This property and the associated optimality conditions will be important for proving the localization properties of proximal steps. These results appear more or less explicitly in *e.g.* (Daniilidis et al., 2006).

Lemma 2.21. *Let $\bar{x}, \bar{y}$ be two points such that g is r prox-regular at $\bar{x}$ for subgradient $\frac{1}{\gamma}(\bar{y} - \bar{x}) \in \partial g(\bar{x})$ with $\gamma \in (0, 1/r)$. Then, $\bar{x}$ is a strong local minimizer of $\rho_{\bar{y}} : u \mapsto g(u) + \frac{1}{2\gamma}\|\bar{y} - u\|^2$,* i.e. *for all x near $\bar{x}$,*

$$\rho_{\bar{y}}(x) \geq \rho_{\bar{y}}(\bar{x}) + \frac{1}{2}\left(\frac{1}{\gamma} - r\right)\|x - \bar{x}\|^2.$$

Proof. Prox-regularity of g at $\bar{x}$ with subgradient $\frac{1}{\gamma}(\bar{y} - \bar{x}) \in \partial g(\bar{x})$ writes

$$g(x) \geq g(\bar{x}) + \frac{1}{\gamma}\langle \bar{y} - \bar{x}, x - \bar{x}\rangle - \frac{r}{2}\|x - \bar{x}\|^2$$

for all x close to $\bar{x}$. The identity $2\langle b-a, c-a\rangle = \|b-a\|^2 + \|c-a\|^2 - \|b-c\|^2$ applied to the previous scalar product yields:

$$g(x) \geq g(\bar{x}) + \frac{1}{2\gamma}\|\bar{y} - \bar{x}\|^2 + \frac{1}{2\gamma}\|x - \bar{x}\|^2 - \frac{1}{2\gamma}\|\bar{y} - x\|^2 - \frac{r}{2}\|x - \bar{x}\|^2,$$

which rewrites

$$\underbrace{g(x) + \frac{1}{2\gamma}\|\bar{y} - x\|^2}_{=\rho_{\bar{y}}(x)} \geq \underbrace{g(\bar{x}) + \frac{1}{2\gamma}\|\bar{y} - \bar{x}\|^2}_{=\rho_{\bar{y}}(\bar{x})} + \frac{1}{2}\left(\frac{1}{\gamma} - r\right)\|x - \bar{x}\|^2.$$

□

2.2.5 Convex case

Let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper lower semi-continuous convex function. From Lemma 2.8, we get that for any $x \in \operatorname{dom} g$ and any $v \in \partial g(x)$, we have

$$g(u) \geq g(x) + \langle v, u - x \rangle \text{ for all } u \in \mathbb{R}^n$$

and adding $\frac{1}{2\gamma}\|u\|^2$ on both sides, we get immediately get that $g + \frac{1}{2\gamma}\|\cdot\|^2$ is bounded below on $\mathbb{R}^n$ for any $\gamma > 0$. Thus, g is prox-bounded with threshold $r_{pb} = 0$. We also immediately get that g is prox-regular with threshold $r = 0$ over the full domain.

Furthermore, $\operatorname{prox}_{\gamma g}$ exists and is unique for any $\gamma > 0$ by Lemma 2.11. Then, Theorem 2.18 simply reduces to the link between first-order optimality conditions of ρ_y and the proximity operator (see (2.4)) and (2.12) gives the firm non-expansivity of the proximal mapping that now holds over $\mathbb{R}^n$.

Theorem 2.22 (Proximity operator of convex functions)**.** *Let g be a convex proper lower semi-continuous function and $\gamma > 0$. Then, for any $y \in \mathbb{R}^n$,*

$$x = \operatorname{prox}_{\gamma g}(y) \quad \Leftrightarrow \quad \frac{y - x}{\gamma} \in \partial g(x).$$

Fruthermore, for any $y, z \in \mathbb{R}^n$, $x = \operatorname{prox}_{\gamma g}(y)$, and $u = \operatorname{prox}_{\gamma g}(z)$, then

$$\|x - u\|^2 \leq \|y - z\|^2 - \|(x - y) - (u - z)\|^2.$$

2.2.6 Examples of explicitly computable proximity operators

In this part, we list some of the most common functions for which the proximity operator can be explicitly computed. In the general case, obtaining a proximity operator requires solving (approximately) an optimization subproblem (which may have better properties than the sole function). While a large literature investigates the properties of inexact proximity operators (see (Martinet, 1970; Rockafellar, 1976) for early works and (Güler, 1992; Lin et al., 2017; Solodov and Svaiter, 2000) for more recent contributions), we will be interested in the *exact*, closed-form, value of the proximity operator in the forthcoming chapters (see especially Chapter 3). A general reference for closed-form proximity operators is the repository http://proximity-operator.net/. Besides, we call the reader's attention to the fact that the explicit computation of the proximity operators displayed below is based on implementable tests that determine the structure of the output.

In terms of analysis, the computation of proximity operators can often be carried on by investigating the points x verifying the first order optimality condition $0 \in \partial g(x) + (x - y)/\gamma$ and comparing their functional values (in terms of ρ_y).

Example 2.23 (Indicators and Projections). For the indicator of some set C ($\iota_C(x) = 0$ if $x \in C$ and $+\infty$ elsewhere), we can directly notice that the proximity operator has to belong to the set (otherwise the inner function is $+\infty$), and thus corresponds to minimizing the distance between the input and a point in C, *i.e.* projecting. ◀

Example 2.24 (ℓ_1 norm). The ℓ_1-norm is defined on $\mathbb{R}^n$ as $\|x\|_1 = \sum_{i=1}^n |x^{[i]}|$. This function is convex (thus prox-regular at every point with $r = 0$). Besides, its proximity

operator admits a closed form expression: for $y \in \mathbb{R}^n$ and $\gamma > 0$, coordinate i writes

$$\operatorname{prox}^{[i]}_{\gamma\|\cdot\|_1}(y) = \begin{cases} y^{[i]} + \gamma & \text{if } y^{[i]} < -\gamma \\ 0 & \text{if } -\gamma \leq y^{[i]} \leq \gamma \\ y^{[i]} - \gamma & \text{if } y^{[i]} > \gamma \end{cases}.$$

This operator is sometimes called soft-thresholding (see *e.g.* (Donoho, 1995)) as it puts to 0 the coordinates corresponding to inputs smaller than γ in absolute value. ◂

Example 2.25 (ℓ_p norms for $p < 1$). The ℓ_0-"norm" is defined on $\mathbb{R}^n$ as $\|x\|_0 = \sum_{i=1}^n \operatorname{ind}_{x^{[i]}}$. This function is non-convex, prox-regular, and its proximity operator still admits a closed form expression: for $y \in \mathbb{R}^n$ and $\gamma > 0$, coordinate i writes

$$\operatorname{prox}^{[i]}_{\gamma\|\cdot\|_0}(y) = \begin{cases} y^{[i]} & \text{if } y^{[i]} < -\gamma \\ 0 & \text{if } -\gamma \leq y^{[i]} \leq \gamma \\ y^{[i]} & \text{if } y^{[i]} > \gamma \end{cases}.$$

In analogy with the case of ℓ_1 norm, this operation is sometimes called hard-thresholding. Whereas the ℓ_0-norm seems a natural way to threshold small values, it is in practice quite unstable compared to the ℓ_1 norm (which can be seen as its convexification).

Between these two extremes, the p-th power of the ℓ_p "norms" for $p \in (0, 1)$ provide potentially useful intermediates in terms of sparsification (Chartrand and Yin, 2016). The proximity operators of these functions are not explicit, except for $p = 1/2$ and $p = 2/3$. Indeed, $\operatorname{prox}^{[i]}_{\gamma\|\cdot\|_p^p}(y)$ is the solution in x of the equation

$$x^{2-p} + y^{[i]} x^{1-p} + 2p = 0$$

which is respectively a cubic equation in $\sqrt{x}$ and a quartic equation in $\sqrt[3]{x}$ for $p = 1/2$ and $p = 2/3$; see (Xu et al., 2012) and (Cao et al., 2013) for the proximity operator derivation. ◂

Example 2.26 (Maximal Value). When $g(x) = \max_i x^{[i]}$, the computation is slightly different since some threshold have to be computed, which is closer to dynamic programming than variational analysis. It is fairly easy to see that the function

$$\rho_y(u) = \max_i u^{[i]} + \frac{1}{2\gamma} \sum_i (u^{[i]} - y^{[i]})^2$$

can be minimized by diminishing the largest value of u, which we note t. We can then divide the coordinates between $\mathcal{I}$ which gather the coordinates for which $y^{[i]} \geq t$, and its complementary. Our proximity candidate is thus $u^{[i]} = t$ for $i \in \mathcal{I}$ and $u^{[i]} = y^{[i]}$ elsewhere. The error for that threshold t is then

$$t + \frac{1}{2\gamma} \sum_{i \in \mathcal{I}} (t - y^{[i]})^2$$

and this cost is minimal if the first order conditions are satisfied *i.e.* if

$$\sum_{i \in \mathcal{I}} (y^{[i]} - t^\star) = \gamma.$$

Then, if $\{i : y^{[i]} \geq t^\star\}$ coincides with $\mathcal{I}$, our candidate solution with $t^\star$ is the sought proximity operator. Otherwise, we have to decrease (resp. increase) t so that

one coordinate more (resp. less) in included in $\mathcal{I}$ depending on $t^\star$. Finally, we obtain that

$$\operatorname{prox}^{[i]}_{\gamma \max}(y) = \min\{t; y^{[i]}\} \text{ where } t \text{ is such that } \sum_{i=1}^{n} \max\{0; y^{[i]} - t\} = \gamma.$$

◀

Example 2.27 (Total variation). In the same spirit as for the maximal value, the total variation in 1D $g(x) = \sum_{i=1}^{n-1} |x^{[i]} - x^{[i+1]}|$ does not have a closed form but can be efficiently computed by dynamic programming. The fastest algorithm to do so seems to be the one of (Condat, 2013). ◀

Example 2.28 (nuclear norm). The nuclear norm is defined on $\mathbb{R}^{n_1 \times n_2}$ as $\|X\|_* = \sum_{i=1}^{\min(n_1,n_2)} \sigma_i$, where σ_i denotes the i-th singular value of X. This function is convex (thus prox-regular at every point with $r = 0$). Its proximity operator admits a closed form expression: for matrix $Y = U\Sigma V^\top$ and step $\gamma > 0$,

$$\operatorname{prox}_{\gamma\|\cdot\|_*}(Y) = U \operatorname{diag}(v_i) V^\top \text{ with } v_i = \operatorname{prox}_{\gamma|\cdot|}(\sigma_i) = \begin{cases} 0 & \text{if } \sigma_i \le \gamma \\ \sigma_i - \gamma & \text{if } \sigma_i > \gamma \end{cases}$$

A remarkable point is that even though two close matrices may have completely different singular vectors, their singular values will be close (this is sometimes called Weyl's lemma, see (Stewart, 1998; Weyl, 1912)). Hence, while a new singular value decomposition has to be computed at each application of the proximity operator of the nuclear norm, the rank of the output will be somewhat stable to small perturbations, which is of utmost importance in the present context. Thus, even though the problem of checking the rank of a matrix can be problematic numerically, the rank of the output of the proximity operator is known by construction. ◀

Example 2.29 (Other matrix functions). As previously considered, the "ℓ_0-equivalent" of the nuclear norm is simply the rank whose proximity operator involves a hard thresholding of the singular values. The ℓ_∞ pendent is the maximal eigenvalue, often encountered in control problems, whose proximity operators is a cutoff of the greatest eigenvalues.

A wide range of functions of matrices thus have explicit proximity operators. More precisely, this is often the case for spectral functions, *i.e.* some function $\tilde{F}$ of the eigenvalues (of real symmetric matrices) since in this case, the proximity operator of $\tilde{F}$ can be directly applied to the eigenvalues (see *e.g.* (Drusvyatskiy and Kempton, 2015, Th. 4.1)). We refer the reader to (Benfenati et al., 2018; Drusvyatskiy and Kempton, 2015; Lewis, 1999) for the analytical tools that enable the derivation of the corresponding proximity operators by examining first-order optimality conditions. ◀

2.3 Numerical optimization methods

Now that we have introduced our core tools, we briefly cover how to use them iteratively to produce numerical methods for minimizing functions, leading to the basic relations that will be ubiquitous in the present manuscript. The results presented here are mainly taken from (Bauschke and Combettes, 2011; Beck, 2017; Bonnans et al., 2006; Bubeck et al., 2015).

2.3.1 The smooth case

Let us start with the problem of solving

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

where f is an L-smooth function (*i.e.* differentiable with a L-Lipschitz continuous gradient). Recalling Lemma 2.6, we obtain our first basic inequalities.

Lemma 2.30. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be an L-smooth function, then for any $x, u \in \mathbb{R}^n$, one has*

$$|f(x) - f(u) - \langle \nabla f(u), x - u\rangle| \leq \frac{L}{2}\|x - u\|^2.$$

In particular, if $x = u - \gamma\nabla f(u)$,

$$f(x) \leq f(u) - \gamma\left(1 - \frac{\gamma L}{2}\right)\|\nabla f(u)\|^2.$$

Thus, taking a gradient step leads to a strict functional decrease ($f(u) < f(x)$) as soon as $\gamma < 2/L$ and $\nabla f(u) \neq 0$. This is the core idea behind the *gradient descent* algorithm. Take $x_0 \in \mathbb{R}^n$ and $\gamma > 0$, the gradient descent algorithm consists in iterating

$$x_{k+1} = x_k - \gamma\nabla f(x_k) \qquad \text{(Gradient descent)}$$

and Lemma 2.30 enables to show the following result.

Theorem 2.31. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be an L-smooth function such that $\inf f > -\infty$. Assume that Gradient descent is run with $0 < \gamma < 2/L$, then $(f(x_k))$ converges and any limit point $\overline{x}$ of (x_k) satisfies $\nabla f(\overline{x}) = 0$.*

Now, if the function is in addition convex, we obtain a more precise control.

Lemma 2.32. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function, then for any $x, u \in \mathbb{R}^n$, one has*

$$f(x) - f(u) \leq \langle \nabla f(x), x - u\rangle - \frac{1}{2L}\|\nabla f(x) - \nabla f(u)\|^2$$
$$\text{and thus } \frac{1}{L}\|\nabla f(x) - \nabla f(u)\|^2 \leq \langle x - u; \nabla f(x) - \nabla f(u)\rangle \leq L\|x - u\|^2$$

which in turn gives that for any $\gamma > 0$,

$$\|x - \gamma\nabla f(x) - (u - \gamma\nabla f(u))\|^2 \leq \|x - u\|^2 - \gamma\left(\frac{2}{L} - \gamma\right)\|\nabla f(x) - \nabla f(u)\|^2.$$

These are the ingredients for the following result where we see that convexity grants us directly pointwise convergence and a functional rate.

Theorem 2.33. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function. Then, the iterates (x_k) generated by Gradient descent with $\gamma = 1/L$ satisfy:*

- *(convergence) $x_k \to x^\star$ for some minimizer $x^\star$ of f;*
- *(rate) $f(x_k) - \min f \leq \dfrac{2L\|x_0 - x^\star\|^2}{k}$ for any minimizer $x^\star$ of f.*

If the function is additionally strongly convex, then the rate changes to a linear one, for a slightly different stepsize. Another important point is that since the function

is now bounded by quadratics from above (by smoothness) and below (by strong convexity), the analysis is mostly done directly on the iterates rather than the functional values. Lemma 2.32 can be improved straightforwardly as follows.

Lemma 2.34 (Bubeck et al. 2015, Lemma 3.11)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a μ-strongly convex L-smooth function, then for any $x, u \in \mathbb{R}^n$, one has*

$$\frac{1}{\mu+L}\|\nabla f(x) - \nabla f(u)\|^2 + \frac{\mu L}{\mu+L}\|x-u\|^2 \leq \langle x-u; \nabla f(x) - \nabla f(u)\rangle \leq L\|x-u\|^2$$

which in turn gives that for any $\gamma > 0$,

$$\|x - \gamma\nabla f(x) - (u - \gamma\nabla f(u))\|^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu+L}\right)\|x-u\|^2 - \gamma\left(\frac{2}{\mu+L} - \gamma\right)\|\nabla f(x) - \nabla f(u)\|^2. \tag{2.13}$$

Then, using that since $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth, its minimizer $x^\star$ satisfies $f(x) - f(x^\star) \leq \frac{L}{2}\|x - x^\star\|^2$, which allow to obtain directly a linear converge from the iterates.

Theorem 2.35. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a μ-strongly convex L-smooth function. Then, the iterates (x_k) generated by Gradient descent with $\gamma = \frac{2}{\mu+L}$ satisfy:*

- *(convergence) $x_k \to x^\star$ for the minimizer $x^\star$ of f;*
- *(rate) $f(x_k) - \min f \leq \left(\frac{\kappa-1}{\kappa+1}\right)^{2k}\|x_0 - x^\star\|^2$ where $\kappa = \frac{L}{\mu} \geq 1$.*

2.3.2 The proximal case

We now consider the problem

$$\min_{x\in\mathbb{R}^n} g(x)$$

where g is not necessarily smooth but admits an exactly computable proximity operator.

In this case, the natural method is the proximal point algorithm:

$$x_{k+1} = \operatorname{prox}_{\gamma g}(x_k) \qquad \text{(Proximal point)}$$

The non-convex case can be difficult to analyze since existence and unicity may not be ensured as covered in Section 2.2. Since there is no other particular inequality to notice, we will only consider the convex case in this subsection.

In the convex case, the first thing to notice is that the fixed points of this algorithm correspond to the minimizers of g, *i.e.* $x^\star$ is a minimizer of g if and only if $x^\star = \operatorname{prox}_{\gamma g}(x^\star)$ (for any $\gamma > 0$); which comes directly from $x^\star = \operatorname{prox}_{\gamma g}(x^\star)$ if and only if $0 \in \partial g(x^\star)$ which is equivalent to $x^\star$ being a minimizer of g since it is convex.

Lemma 2.36 ((Bauschke and Combettes, 2011, Prop. 12.27))**.** *Let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a convex lower semi-continuous proper function, then for any $x, u \in \mathbb{R}^n$*

$$g(\operatorname{prox}_{\gamma g}(x)) + \frac{1}{\gamma}\|\operatorname{prox}_{\gamma g}(x) - x\|^2 \leq g(u) + \frac{1}{\gamma}\|u-x\|^2 - \frac{1}{\gamma}\|\operatorname{prox}_{\gamma g}(x) - u\|^2$$

and

$$\begin{aligned} &\|\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma g}(u)\|^2 \leq \langle x - u, \mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma g}(u)\rangle \\ \Leftrightarrow &\|\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma g}(u)\|^2 \leq \|x - u\|^2 - \|x - \mathrm{prox}_{\gamma g}(x) - u + \mathrm{prox}_{\gamma g}(u)\|^2 \end{aligned} \tag{2.14}$$

Then, the analysis of the proximal point can be carried quite directly.

Theorem 2.37. *Let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a convex lower semi-continuous proper function. Then, the Proximal point method with $\gamma > 0$ verifies $g(x_{k+1}) \leq g(x_k)$ and*

$$g(x_k) - \min g \leq \frac{\|x^\star - x_0\|^2}{2\gamma k}$$

for any minimizer $x^\star$ of g.

Remark 2.38 (Multiple functions). When the objective is a sum of functions $g = \sum_{i=1}^M g_i$, the use of the proximity operator is less direct since the proximity operator of the sum cannot be computed in general from the individual proximity operators. In order to deal with that case, two techniques can be combined:

- Lifting the space from $\mathbb{R}^n$ to $\mathbb{R}^{n\times M}$ and using the equivalence of

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^M g_i(x) \text{ and } \min_{\tilde{x}=(x_1,..,x_M)\in\mathbb{R}^{n\times M}} \underbrace{\sum_{i=1}^M g_i(x_i)}_{\tilde{g_1}(\tilde{x})} + \underbrace{\iota_{x_1=x_2=..=x_M}(\tilde{x})}_{\tilde{g_2}(\tilde{x})};$$

- Using a splitting method to find a minimizer of $\tilde{g_1} + \tilde{g_2}$.

For more details, see the review (Condat et al., 2019). ◂

2.3.3 The composite case

Finally, putting the two above problems together, we now address the minimization of the sum of a smooth function and a nonsmooth one:

$$\min_{x\in\mathbb{R}^n} F(x) = f(x) + g(x)$$

where f is accessible though its gradient (but its proximity operator is in general out of reach) but the proximity operator of g can be computed.

In order to decouple these two functions, *splitting* methods have been developed. They consist in finding a critical point of F by solving the fixed-point iteration

$$\begin{aligned} & 0 \in \gamma\nabla f(x_k) + \gamma\partial g(x_{k+1}) + x_{k+1} - x_k \\ \Leftrightarrow\quad & 0 \in \partial g(x_{k+1}) + \frac{1}{\gamma}\left(x_{k+1} - (x_k - \gamma\nabla f(x_k))\right) \\ \Leftrightarrow\quad & x_{k+1} = \mathrm{prox}_{\gamma g}\left(x_k - \gamma\nabla f(x_k)\right) \end{aligned}$$

and thus consists in alternating a proximal step and a gradient step.[11]

The proximal gradient algorithm then consists in iterating

$$x_{k+1} = \mathrm{prox}_{\gamma g}\left(x_k - \gamma\nabla f(x_k)\right) \qquad \text{(Proximal gradient)}$$

[11] see Section 2.2 for the conditions behind the last equivalence in the non-convex case.

for some $\gamma > 0$ and starting point x_0.

It is worth noticing that this composition can actually be seen as the minimization of a first-order approximation of f plus g. Indeed:

$$\begin{aligned} x_{k+1} &= \operatorname{prox}_{\gamma g}\left(x_k - \gamma \nabla f(x_k)\right) \\ &= \operatorname{argmin}_{u \in \mathbb{R}^n} \left\{ g(u) + \frac{1}{2\gamma}\|u - x_k + \gamma \nabla f(x_k)\|^2 \right\} \\ &= \operatorname{argmin}_{u \in \mathbb{R}^n} \left\{ f(x_k) + \langle u - x_k, \nabla f(x_k)\rangle + g(u) + \frac{1}{2\gamma}\|u - x_k\|^2 \right\}. \end{aligned}$$

By carefully expressing the fact the x_{k+1} is a minimizer of the function inside the braces and using the functions properties, we get the following inequalities.

Lemma 2.39. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-smooth function, and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a lower semi-continuous proper function. Take $\gamma > 0$ and define the proximal gradient operator as $\mathsf{T}(x) = \operatorname{prox}_{\gamma g}(x - \gamma \nabla f(x))$ for all $x \in \mathbb{R}^n$.*
Then for any $x \in \mathbb{R}^n$, supposing that $\mathsf{T}(x)$ is well defined,

$$F(\mathsf{T}(x)) \leq F(x) + \frac{1}{2}\left(L - \frac{1}{\gamma}\right)\|\mathsf{T}(x) - x\|^2.$$

If f is convex, for any $x, u \in \mathbb{R}^n$, supposing that $\mathsf{T}(x)$ is well defined,

$$F(\mathsf{T}(x)) \leq F(u) + \frac{1}{2\gamma}\|u - x\|^2 + \frac{1}{2}\left(L - \frac{1}{\gamma}\right)\|\mathsf{T}(x) - x\|^2.$$

If g is convex, for any $x \in \mathbb{R}^n$,

$$F(\mathsf{T}(x)) \leq F(x) + \frac{1}{2}\left(L - \frac{2}{\gamma}\right)\|\mathsf{T}(x) - x\|^2.$$

Finally, if both f and g are convex, for any $x, u \in \mathbb{R}^n$,

$$F(\mathsf{T}(x)) \leq F(u) + \frac{1}{2\gamma}\|u - x\|^2 - \frac{1}{2\gamma}\|u - \mathsf{T}(x)\|^2 + \frac{1}{2}\left(L - \frac{1}{\gamma}\right)\|\mathsf{T}(x) - x\|^2 \tag{2.15}$$

and provided that $\gamma \in (0, \frac{2}{L})$,

$$\|\mathsf{T}(x) - \mathsf{T}(u)\|^2 \leq \|x - u\|^2 - \frac{1-\nu}{\nu}\|(\mathrm{I} - \mathsf{T})(x) - (\mathrm{I} - \mathsf{T})(u)\|^2$$

with $\nu = \frac{2}{1+2\min\{1, 1/(\gamma L)\}}$.

In the fully convex case, this enables us to get the following result.

Theorem 2.40. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function and let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a convex lower semi-continuous proper function. Then, the Proximal gradient method with $\gamma \in (0, 1/L]$ verifies $F(x_{k+1}) \leq F(x_k)$ and*

$$F(x_k) - \min F \leq \frac{\|x^\star - x_0\|^2}{2\gamma k}$$

for any minimizer $x^\star$ of F.

When f is additionally μ-strongly convex, combining (2.13) and (2.14), enables to show exactly the same results as for Gradient descent.

Finally, it is interesting to note that the difference between the input and output of a proximity operator is linked to the subdifferential of the function.

Lemma 2.41 (Bolte et al. 2015, Prop. 13)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-smooth function, and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a lower semi-continuous proper function. Take $\gamma > 0$ and define the proximal gradient operator as $\mathsf{T}(x) = \mathrm{prox}_{\gamma g}(x - \gamma \nabla f(x))$ for all $x \in \mathbb{R}^n$. Then, for all $x \in \mathbb{R}^n$,*

$$\mathrm{dist}(0, \partial F(\mathsf{T}(x))) \leq \frac{\gamma L + 1}{\gamma} \|x - \mathsf{T}(x)\|.$$

We end this section with the words of Cauchy (Cauchy et al., 1847): *"I'll restrict myself here to outlining the principles underlying [my method], with the intention to come again over the same subject".*[12]

[12] In the original text: *" Je me bornerai pour l'instant à indiquer les principes sur lesquels [ma méthode] se fonde, me proposant de revenir avec plus de détails sur le même sujet."*. The translation and reference is due to Claude Lemaréchal (Lemaréchal, 2012).

2.4 Riemannian manifolds

We introduce below the tools of Riemannian optimization that will be used in this manuscript. In particular, we restrict ourselves to C^2 submanifolds of $\mathbb{R}^n$. We refer the reader to (Absil et al., 2009) and (Boumal, 2020) for more extensive presentations.

2.4.1 Manifolds

A subset $\mathcal{M}$ of $\mathbb{R}^n$ is said to be a *p-dimensional C^2 submanifold* of $\mathbb{R}^n$ around $x \in \mathcal{M}$ if there exists a local parameterization of $\mathcal{M}$ around x, that is, a C^2 function $\varphi : \mathbb{R}^p \to \mathbb{R}^n$ such that φ realizes a local homeomorphism between a neighborhood of $0 \in \mathbb{R}^p$ and a neighborhood of $x \in \mathcal{M}$ and the derivative of φ at $\varphi^{-1}(x) = 0$ is injective. A p-dimensional C^2-submanifold of $\mathbb{R}^n$ can alternatively be defined via a local equation, that is a C^2 function $\Phi : \mathbb{R}^n \to \mathbb{R}^{n-p}$ with a surjective derivative at $\bar{x} \in \mathcal{M}$ that satisfies for all x close enough to $\bar{x}$, $x \in \mathcal{M} \Leftrightarrow \Phi(x) = 0$.

A basic tool to investigate approximations on manifolds is notion of the *smooth curves*. A smooth curve on $\mathcal{M}$ is a C^2 application $c : I \subset \mathbb{R} \to \mathcal{M} \subset \mathbb{R}^n$, where I is an open interval containing 0. At each point $x \in \mathcal{M}$, the *tangent space*, noted $T_x\mathcal{M}$, can be defined as the velocities of all smooth curves passing by x at 0:

$$T_x\mathcal{M} := \{c'(0) \mid c : I \to \mathcal{M} \text{ is a smooth curve around } 0 \text{ and } c(0) = x\}.$$

The tangent space is a p-dimensional Euclidean space containing *tangent vectors*. Thus, each tangent space $T_x\mathcal{M}$ is equipped with a scalar product $\langle \cdot, \cdot \rangle_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$, and the associated norm $\| \cdot \|_x$. In many cases, the tangent metric varies smoothly with x, making the manifold *Riemannian*. In this following, we use the ambient space scalar product to define the scalar product on tangent spaces; we will thus drop the subscript in the tangent scalar product and norm notations when there is no confusion possible. We will denote by proj_x the orthogonal projector from $\mathbb{R}^n$ to $T_x\mathcal{M}$. Finally, we also define the *normal space* at $x \in \mathcal{M}$ as the orthogonal space to $T_x\mathcal{M}$ and the *tangent bundle* as $T\mathcal{B} := \bigcup_{x \in \mathcal{M}} (x, T_x\mathcal{M})$.

A *metric* on $\mathcal{M}$ can be defined as the minimal length over all curves joining two points $x, u \in \mathcal{M}$, ie. $\mathrm{dist}_{\mathcal{M}}(x, u) = \inf_{c \in C_{x,u}} \int_0^1 \|c'(t)\|_{c(t)} \mathrm{d}t$, where $C_{x,u}$ is the set of $[0, 1] \to \mathcal{M}$ smooth curves c such that $c(0) = x$, $c(1) = u$. The minimizing curves

generalize the notion of straight line between two points to manifolds. The constant speed parametrization of any minimizing curves is called a *geodesic*.

Remark 2.42. Both tangent and normal spaces of at $x \in \mathcal{M}$ admit explicit expression from local parametrizations φ or local equations Φ defining $\mathcal{M}$:

$$T_x\mathcal{M} = \text{Im } \mathrm{D}_\varphi(0) = \text{Ker } \mathrm{D}_\Phi(x) \qquad N_x\mathcal{M} = \text{Ker } \mathrm{D}_\varphi(0)^* = \text{Im } \mathrm{D}_\Phi(x)^*$$

◀

2.4.2 Functions on manifolds

The notion of differential and gradient can be extended to functions defined on manifolds. Let $F : \mathcal{M} \to \mathbb{R}$. The *Riemannian differential* of F at x is the linear operator $\mathrm{D}F(x) : T_x\mathcal{M} \to \mathbb{R}$ defined by $\mathrm{D}F(x)[\eta] := \frac{\mathrm{d}}{\mathrm{d}t} F \circ c(t)\big|_{t=0}$, where c is a smooth curve such that $c(0) = x$ and $c'(0) = \eta$. In turn, the *Riemannian gradient* $\text{grad } F(x)$ is the unique vector of $T_x\mathcal{M}$ such that, for any tangent vector η, $\mathrm{D}F(x)[\eta] = \langle \text{grad } F(x), \eta \rangle$.

If $\text{grad } F(x)$ exists at any point of $\mathcal{M}$, then F said to be differentiable. In that case, a first order Taylor development can be formulated. Let $x \in \mathcal{M}$, $\eta \in T_x\mathcal{M}$ and c denote a smooth curve passing by x, with velocity η at 0 (abbreviated "passing by x, η" in the following). Then, for t near 0,

$$F \circ c(t) = F(x) + t\langle \text{grad } F(x), \eta \rangle + o(t).$$

Before defining second order objects, we need a notion of derivation for vector fields and of acceleration for curves. Consider a curve $c : I \to \mathcal{M}$ and a smooth vector field Z on that curve, that is a smooth map such that $Z(t) \in T_{c(t)}\mathcal{M}$ for $t \in I$. The *covariant derivative* of Z on the curve c, denoted $\frac{\mathrm{D}}{\mathrm{d}t} Z : I \to T\mathcal{B}$, is defined by $\frac{\mathrm{D}}{\mathrm{d}t} Z(t) := \text{proj}_{c(t)} Z'(t)$, where $Z'(t)$ denotes the derivative in the ambient space $\mathbb{R}^n$ and proj_x corresponds to the orthogonal projector from $\mathbb{R}^n$ to $T_x\mathcal{M}$. The *acceleration* of a curve c is defined as the covariant derivative of its velocity: $c''(t) := \frac{\mathrm{D}}{\mathrm{d}t} c'(0)$. It amounts to the tangential component of the second derivative of the curve in $\mathbb{R}^n$.

The *Riemannian hessian* of F at x along η is the linear operator $\text{Hess } F(x) : T_x\mathcal{M} \to T_x\mathcal{M}$ defined by the relation $\text{Hess } F(x)[\eta] := \frac{\mathrm{D}}{\mathrm{d}t} \text{grad } F(c(t))\big|_{t=0}$, where c is a smooth curve such that $c(0) = x$, $c'(0) = \eta$. The Riemannian hessian may be defined equivalently by the relation $\langle \text{Hess } F(x)[\eta], \eta \rangle = \frac{\mathrm{d}^2}{\mathrm{d}t^2} F \circ \gamma(t)\Big|_{t=0}$, where γ is a geodesic curve such that $\gamma(0) = x$, $\gamma'(0) = \eta$. A second order Taylor development can now be formulated. Let $x \in \mathcal{M}$, $\eta \in T_x\mathcal{M}$, and c be a smooth curve such that $c(0) = x$, $c'(0) = \eta$. Then, for t near 0,

$$F \circ c(t) = F(x) + t\langle \text{grad } F(x), \eta \rangle + \frac{t^2}{2}\langle \text{Hess } F(x)[\eta], \eta \rangle + \frac{t^2}{2}\langle \text{grad } F(x), c''(0) \rangle + o(t^2).$$

If $F : \mathcal{M} \to \mathbb{R}$ has a smooth extension on $\mathbb{R}^n$, the Riemannian gradient and Hessian can be computed from their Euclidean counterparts: for a smooth function $\bar{F} : \mathbb{R}^n \to \mathbb{R}$ that coincides with F on $\mathcal{M}$, we have

$$\text{grad } F(x) = \text{proj}_x(\nabla \bar{F}(x)), \tag{2.16}$$

and, for $\bar{G} : \mathbb{R}^n \to \mathbb{R}^n$ a smooth mapping that coincides with $\text{grad } F$ on $\mathcal{M}$, we have

$$\text{Hess } F(x)[\eta] = \text{proj}_x\left(\mathrm{D}\bar{G}(x)[\eta]\right). \tag{2.17}$$

2.4.3 Optimality on manifolds

Lemma 2.43 (Optimality conditions)**.** *Consider a manifold $\mathcal{M}$ embedded in $\mathbb{R}^n$, a point $\bar{x}$ of $\mathcal{M}$ and a function F defined on a neighborhood of $\bar{x}$ in $\mathcal{M}$ for which $\bar{x}$ is a strong minimizer ie. for all x near $\bar{x}$ in M,*

$$F()x) \geq F(\bar{x}) + \frac{c}{2}\|x - \bar{x}\|^2.$$

Then, there holds

$$\operatorname{grad} F(\bar{x}) = 0 \quad \textit{and} \quad \operatorname{Hess} F(\bar{x}) \succeq cI.$$

Proof. Let $\eta \in T_{\bar{x}}\mathcal{M}$ and denote γ a geodesic going through $\bar{x}$ with velocity η at $t = 0$. The quadratic growth assumption can be applied at $x = \gamma(t)$, which allows to write

$$\frac{1}{t}(F \circ \gamma(t) - F \circ \gamma(0)) \geq \frac{c}{2}\left\|\frac{\gamma(t) - \gamma(0)}{\sqrt{t}}\right\|^2.$$

Taking the limit $t \to 0$ yields $\langle \operatorname{grad} F(\bar{x}), \eta\rangle \geq 0$. The same reasoning holds with $x = \gamma(-t)$ and yields the converse inequality, so that $\langle \operatorname{grad} F(\bar{x}), \eta\rangle = 0$.

Besides, summing the quadratic growth conditions applied at $\gamma(t)$ and $\gamma(-t)$ provides

$$\frac{1}{t^2}(F \circ \gamma(t) - 2F \circ \gamma(0) + F \circ \gamma(-t)) \geq \frac{c}{2}\left\|\frac{\gamma(t) - \gamma(0)}{t}\right\|^2 + \frac{c}{2}\left\|\frac{\gamma(-t) - \gamma(0)}{t}\right\|^2.$$

Taking the limit as $t \to 0$ yields $\langle \operatorname{Hess} F(\bar{y})[\eta], \eta\rangle \geq c\|\eta\|^2$. As η is picked arbitrarily in $T_{\bar{x}}\mathcal{M}$, the results are obtained. □

2.4.4 Retractions

In practice, iterative methods require a way to produce curves on $\mathcal{M}$ given a point x and tangent vector η. A geodesic curve passing at x, η, while attractive as the generalization of the straight line, has a prohibitive computational cost. It thus common to replace those with first or second-order approximations.

Definition 2.44. A *retraction* on a manifold $\mathcal{M}$ is a smooth map $\mathrm{R} : T\mathcal{B} \to \mathcal{M}$ such that

i) $\mathrm{R}_x(0) = x$, and

ii) $\mathrm{D}\,\mathrm{R}_x(0) : T_x\mathcal{M} \to T_x\mathcal{M}$ is the identity map: $\mathrm{D}\,\mathrm{R}_x(0)[\eta] = \eta$,

where, for each $x \in \mathcal{M}$, $\mathrm{R}_x : T_x\mathcal{M} \to \mathcal{M}$ is defined as the restriction of R at x, so that $\mathrm{R}_x(\eta) = \mathrm{R}(x, \eta)$. A *second-order retraction* is a retraction R such that, for all $(x, \eta) \in T\mathcal{B}$, the curve $c(t) = \mathrm{R}_x(t\eta)$ has zero acceleration at 0: $c''(0) = 0$.

We will consider the retractions to be of second order in the remaining of the manuscript.

With a second-order retraction, $t \mapsto \mathrm{R}_x(t\eta)$ provides a practical and relatively cheap curve passing by x, η at 0, giving the following second order approximation:

$$F \circ \mathrm{R}_x(t\eta) = F(x) + t\langle \operatorname{grad}\ F(x), \eta\rangle + \frac{t^2}{2}\langle \operatorname{Hess}\ F(x)[\eta], \eta\rangle + o(t^2). \tag{2.18}$$

2.4.5 Euclidean spaces and manifolds, back and forth

In this document, we will use Riemannian and Euclidean methods in harmony to try and take the best of both worlds; we thus need some control over the passage from the Euclidean to the Riemannian geometry.

Lemma 2.45. *Consider a point $\bar{x}$ of a Riemannian manifold $\mathcal{M}$, equipped with a retraction R such that $\mathrm{R}_{\bar{x}}$ is C^2. For any $\varepsilon > 0$, there exists a neighborhood $\mathcal{U}$ of $\bar{x}$ in $\mathcal{M}$ such that, for all $x \in \mathcal{U}$,*

$$(1-\varepsilon)\mathrm{dist}_{\mathcal{M}}(x,\bar{x}) \leq \|\mathrm{R}_{\bar{x}}^{-1}(x)\| \leq (1+\varepsilon)\mathrm{dist}_{\mathcal{M}}(x,\bar{x}).$$

where $\mathrm{R}_{\bar{x}}^{-1} : \mathcal{M} \to T_{\bar{x}}\mathcal{M}$ is the smooth inverse of $\mathrm{R}_{\bar{x}}$ defined locally around $\bar{x}$.

Proof. The retraction at $\bar{x}$ can be inverted locally around 0. Indeed, as $\mathrm{D}\,\mathrm{R}_{\bar{x}}(0_{T_{\bar{x}}\mathcal{M}}) = I$ is invertible and $\mathrm{R}_{\bar{x}}$ is C^2, the implicit function theorem provides the existence of a C^2 inverse function $\mathrm{R}_{\bar{x}}^{-1} : \mathcal{M} \to T_{\bar{x}}\mathcal{M}$ defined locally around $\bar{x}$. Furthermore, one shows by differentiating the relation $\mathrm{R}_{\bar{x}} \circ \mathrm{R}_{\bar{x}}^{-1}$ that the differential of $\mathrm{R}_{\bar{x}}^{-1}$ at $\bar{x}$ is the identity.

We consider the function $F : \mathcal{M} \to \mathbb{R}$ defined by $F(x) = \|\log_{\bar{x}}(x)\| - \|\mathrm{R}_{\bar{x}}^{-1}(x)\|$.[13] Clearly $F(\bar{x}) = 0$, and $\mathrm{D}\,F(\bar{x}) = 0$ as the differentials of both $\mathrm{R}_{\bar{x}}^{-1}$ and logarithm at $\bar{x}$ are the identity. In local coordinates $\hat{x} = \log_{\bar{x}} x$ around $\bar{x}$, F is represented by the function $\hat{F} = F \circ \exp_{\bar{x}} : T_{\bar{x}}\mathcal{M} \to \mathbb{R}$. As $\hat{F}(\hat{\bar{x}}) = 0$, $\mathrm{D}\,\hat{F}(\hat{\bar{x}}) = 0$ and $\hat{F}$ is C^2, there exists some $C > 0$ such that, over a neighborhood $\hat{\mathcal{U}}$ of $\hat{\bar{x}}$,

$$-C\|\hat{x} - \hat{\bar{x}}\|^2 \leq \hat{F}(\hat{x}) \leq C\|\hat{x} - \hat{\bar{x}}\|^2.$$

[13] The exponential and logarithmic maps are respectively a retraction and its inverse associated with a particular geodesic, see (Boumal, 2020, Chap. 10.2)

For any $\varepsilon > 0$, by taking a small enough neighborhood $\hat{\mathcal{U}}' \subset \hat{\mathcal{U}}$, there holds

$$-\varepsilon\|\hat{x} - \hat{\bar{x}}\| \leq \hat{F}(\hat{x}) \leq \varepsilon\|\hat{x} - \hat{\bar{x}}\|.$$

Thus for all x in $\mathcal{U} = \mathrm{R}_{\bar{x}}(\hat{\mathcal{U}}')$,

$$-\varepsilon\|\log_{\bar{x}}(x)\| \leq \|\log_{\bar{x}}(x)\| - \|\mathrm{R}_{\bar{x}}^{-1}(x)\| \leq \varepsilon\|\log_{\bar{x}}(x)\|,$$

as $\hat{x} = \log_{\bar{x}}(x)$, $\hat{\bar{x}} = 0$. The result's equivalent form is obtained using that $\mathrm{dist}_{\mathcal{M}}(x,\bar{x}) = \|\hat{x} - \hat{\bar{x}}\| = \|\log_{\bar{x}}(x)\|$. □

We recall a slightly specialized version of (Miller and Malick, 2005, Th. 2.2), which is essentially the application of the implicit function theorem around a point of a manifold.

Proposition 2.46 (Tangential retraction)**.** *Consider a p-dimensional C^k-submanifold $\mathcal{M}$ of $\mathbb{R}^n$ around a point $\bar{x} \in \mathcal{M}$. The mapping $\mathrm{R} : T\mathcal{B} \to \mathcal{M}$, defined for $(x,\eta) \in T\mathcal{B}$ near $(\bar{x},0)$ by $\mathrm{proj}_x(\mathrm{R}(x,\eta)) = \eta$ defines a second-order retraction near $(\bar{x},0)$. The pointwise retraction, defined as $\mathrm{R}_x = \mathrm{R}(x,\cdot)$, is locally invertible with inverse $\mathrm{R}_x^{-1} = \mathrm{proj}_x$.*

Proof. Let $\Psi : \mathbb{R}^n \to \mathbb{R}^{n-p}$ denote a C^k function defining $\mathcal{M}$ around $\bar{x}$: for all x close enough to $\bar{x}$, there holds $x \in \mathcal{M} \Leftrightarrow \Psi(x) = 0$, and $\mathrm{D}\,\Psi(x)$ is surjective. Consider the equation $\Phi(x,\eta_t,\eta_n) = 0$ around $(\bar{x},0,0)$, with

$$\begin{aligned} \Phi : \{x,\eta_t,\eta_n : x \in \mathcal{M}, \eta_t \in T_x\mathcal{M}, \eta_n \in N_x\mathcal{M}\} &\to \mathbb{R} \\ x,\eta_t,\eta_n &\mapsto \Psi(x+\eta_t+\eta_n). \end{aligned}$$

The partial differential $\mathrm{D}_{\eta_n} \Phi(\bar{x}, 0, 0)$ is, for $\xi_n \in N_{\bar{x}}\mathcal{M}$,

$$\mathrm{D}_{\eta_n} \Phi(\bar{x}, 0, 0)[\xi_n] = \mathrm{D}\,\Psi(\bar{x})[\xi_n].$$

Since $\bar{x} \in \mathcal{M}$, $\mathrm{D}_{\eta_n} \Phi(\bar{x}, 0, 0)$ is surjective from $N_{\bar{x}}\mathcal{M}$ to $\mathbb{R}^{n-p}$ so its a bijection. The implicit function theorem provides the existence of neighborhoods $\mathcal{N}_{\bar{x}}^1 \subset \mathcal{M}$, $\mathcal{N}_0^2 \subset \cup_{x\in\mathcal{M}} T_x\mathcal{M}$, $\mathcal{N}_0^3 \subset \cup_{x\in\mathcal{M}} N_x\mathcal{M}$ and a unique C^k function $\eta_n : \mathcal{N}_{\bar{x}}^1 \times \mathcal{N}_0^2 \to \mathcal{N}_0^3$ such that, for all $x \in \mathcal{N}_{\bar{x}}^1$, $\eta_t \in \mathcal{N}_0^2$ and $\eta_n \in \mathcal{N}_0^3$, $\eta_n(\bar{x}, 0) = 0$ and

$$\Phi(x, \eta_t, \eta_n(x, \eta_t)) = 0 \Leftrightarrow x + \eta_t + \eta_n(x, \eta_t) \in \mathcal{M}.$$

It also provides an expression for the partial derivative of η_n at $(x, 0)$ along η_t: for $\xi_t \in T_x\mathcal{M}$,

$$\mathrm{D}_{\eta_t} \eta_n(x, 0)[\xi_t] = -\left[\mathrm{D}_{\eta_n} \Phi(x, 0, 0)\right]^{-1} \mathrm{D}_{\eta_t} \Phi(x, 0, 0)[\xi_t].$$

As noted before, $\mathrm{D}_{\eta_n} \Phi(x, 0, 0)$ is one-to-one since $x \in \mathcal{M}$. Besides, $\mathrm{D}_{\eta_t} \Phi(x, 0, 0) = \mathrm{D}\,\Phi(x)[\xi_t] = 0$ since $T_x\mathcal{M}$ identifies as the kernel of $\mathrm{D}\,\Phi(x)$. Thus $\mathrm{D}_{\eta_t} \eta_n(x, 0) = 0$.

Now, define a map $\mathrm{R} : \mathcal{N}_{\bar{x}}^1 \times \mathcal{N}_0^2 \to \mathcal{M}$ by $\mathrm{R}(x, \eta_t) = x + \eta_t + \eta_n(x, \eta_t)$. This map has degree of smoothness C^k since η_n is C^k, satisfies $\mathrm{R}(x, 0) = x$ since $\eta_n(x, 0) = 0$ and satisfies $\mathrm{D}_{\eta_t} \eta_n(x, 0) = I + \mathrm{D}_{\eta_t}(x, 0) = I$. Thus R defines a retraction on a neighborhood of $(\bar{x}, 0)$.

We turn to show the second-order property of R. Consider the smooth curve c defined as $c(t) = \mathrm{R}(x, t\eta)$ for some $x \in \mathcal{N}_{\bar{x}}^1$, $\eta_t \in T_x\mathcal{M} \cap \mathcal{N}_0^2$. Its first derivative writes

$$c'(t) = \eta + \mathrm{D}_{\eta_t} \eta_n(x, t\eta)[\eta] = \eta.$$

As $c'(t) \in T_x\mathcal{M}$, and we consider a Riemannian submanifold of $\mathbb{R}^n$, the acceleration of the curve c is obtained by computing the derivative of $c'(\cdot)$ in the ambient space and then projecting onto $T_x\mathcal{M}$. Thus $c''(t) = 0$ and in particular, $c''(0) = 0$ which makes R a second-order retraction. □

Lemma 2.47. *Consider a point $\bar{x}$ of a Riemannian manifold $\mathcal{M}$. For any $\varepsilon > 0$, there exists a neighborhood $\mathcal{U}$ of $\bar{x}$ in $\mathcal{M}$ such that, for all $x \in \mathcal{U}$,*

$$(1 - \varepsilon)\mathrm{dist}_{\mathcal{M}}(x, \bar{x}) \leq \|x - \bar{x}\| \leq (1 + \varepsilon)\mathrm{dist}_{\mathcal{M}}(x, \bar{x}),$$

where $\|x - \bar{x}\|$ is the Euclidean distance in the ambient space.

Proof. Let $\bar{x}$, x denote two close points on $\mathcal{M}$. Consider the tangential retraction introduced in Proposition 2.46. As a retraction, it satisfies:

$$\mathrm{R}_{\bar{x}}(\eta) = \mathrm{R}_{\bar{x}}(0) + \mathrm{D}\,\mathrm{R}_{\bar{x}}(0)[\eta] + O(\|\eta\|^2) = \bar{x} + O(\|\eta\|^2).$$

Taking $x = \mathrm{R}_{\bar{x}}(\eta)$ allows to write $x = \bar{x} + O(\|\mathrm{R}_{\bar{x}}^{-1}(x)\|^2)$, so that for any small $\varepsilon_1 > 0$, there exists a small enough neighborhood $\mathcal{U}_1 \subset \mathcal{U}$ of $\bar{x}$ in $\mathcal{M}$ such that

$$(1 - \varepsilon_1)\|\mathrm{R}_{\bar{x}}^{-1}(x)\| \leq \|x - \bar{x}\| \leq (1 + \varepsilon_1)\|\mathrm{R}_{\bar{x}}^{-1}(x)\|.$$

By Lemma 2.45, for $\varepsilon_2 > 0$ small enough, there exists a neighborhood $\mathcal{U}_2 \subset \mathcal{U}$ of $\bar{x}$ such that,

$$(1 - \varepsilon_2)\mathrm{dist}_{\mathcal{M}}(x, \bar{x}) \leq \|\mathrm{R}_{\bar{x}}^{-1}(x)\| \leq (1 + \varepsilon_2)\mathrm{dist}_{\mathcal{M}}(x, \bar{x}).$$

By choosing ε_1, ε_2 such that $1 - \varepsilon = (1 - \varepsilon_1)(1 - \varepsilon_2)$, these two estimates can be combined to get the result. □

PART A

STRUCTURE IDENTIFICATION IN DATA SCIENCE: THEORY, PRACTICE, AND ACCELERATION OF OPTIMIZATION METHODS

Many problems in Data Science (regression, classification, clustering, etc.) lead to the minimization of some *risk* function that measures the adequation between a model and the data. However, when the number of parameters of the model becomes large and the difficulty of the problem increases, the risk minimization gets harder and the stability of the obtained model is degraded.

In order to overcome this issue, a popular solution is to introduce a *prior* on the *structure* of the model. For instance, we may want the obtained model to be sparse or low-rank so that the number of (non-null) parameters of the model is not too large; which is helpful in terms of interpretability, generalization, and for the minimization itself.

In this part, we study how certain optimization methods can produce iterates that recover partially or exactly this structure. Then, we show how this information can be harnessed to numerically accelerate optimization algorithms.

Motivation: Regularized Learning Problems

As mentioned in the introduction, many task of interest in Data Science lead to an empirical risk minimization problem

$$\min_{x \in \Theta} \hat{R}(P_x) = \frac{1}{m} \sum_{i=1}^{m} \ell(b_i, P_x(\mathrm{a}_i)) \tag{ERM}$$

which is usually smooth (by modelling or by approximation).

On the contrary, regularization implies resorting to nonsmooth functions. Indeed, there is a strong link between non-differentiability points and structure stability,[14] which made the use of nonsmooth regularizers ubiquitous in machine learning and signal processing for the nice *recovery* or *consistency* properties that they induce; see e.g. (Vaiter et al., 2015) and recall Fig. 1.1.

[14] This will be explored in Chapter 3.

In general, the structure priors in machine learning (*e.g.* sparsity, fixed rank, piecewise constant, etc.) are subsets of $\mathbb{R}^n$ that are rather easy to express and to project on. As mathematical objects, to comply with a vast part of the literature, we shall see them as smooth manifolds. A regularizer can then be defined as a nonsmooth function

Ω whose non-differentiability points coincide with the sough structure (*e.g.* the ℓ_1-norm for sparsity, the nuclear norm for low-rank, the total variation for piecewise constant signals, etc.). This leads to a regularized empirical risk minimization problem:

$$\min_{x\in\mathbb{R}^n} \underbrace{\frac{1}{m}\sum_{i=1}^{m} \ell(b_i, P_x(\mathrm{a}_i))}_{\substack{=:f(x)\\ \text{minimizes the risk } \hat{R}(P_x)}} + \underbrace{\lambda\,\Omega(x)}_{\substack{=:g(x)\\ \text{enforces structure}}} \qquad \text{(Regularized ERM)}$$

This problem has a composite form, involving two different parts: f which is linked to the risk itself, and g which only promotes structure, with a hyperparameter $\lambda > 0$ controlling the balance between risk minimization and structure enforcement. In this document, we will consider λ as a constant fixed by the user.[15] We refer to (Bach et al., 2012; Candes et al., 2008; Combettes and Pesquet, 2011; Vaiter et al., 2017) among a vast literature, for references about the theoretical and practical aspects of regularization.

[15] In some situations, the exact structure of the unknown original can be recovered with some probability by appropriately choosing it; see *e.g.* (Bach, 2008; Candès et al., 2006; Fadili et al., 2019; Zhao and Yu, 2006)

Finally, the nonsmooth structure-enforcing part $g = \lambda\Omega$ is generally chosen so that its proximity operator can be computed easily (with a closed form or efficiently); see Section 2.2. This motivates our study of proximal optimization methods in view of their structure recovery abilities.

Nonsmooth Optimization & Structure identification

In this part, we will consider problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x) \qquad (\mathcal{P}_\mathrm{A})$$

where f is a smooth function and g is nonsmooth.[16]

[16] The precise assumptions will vary from one chapter to another but the structure will stay the same.

This problem is nonsmooth *but*, in our applications of interest, it is structured:

- S_1 the points of non-differentiability of g follow a known structure (this is our prior), and prox_g can be exactly computed;
- S_2 the properties of the problem (smoothness, conditioning) are often improved when restricted to the local structure.

We will thus investigate in this part how these two properties can be harnessed to provide theoretical structure identification results and numerical acceleration schemes for optimization methods.

This course of action is rather original. Indeed, the investigation of functions whose non-differentiability points and proximity operators are known beforehand was scarcely studied in the optimization literature, where structure was often known/reached inexactly. Nevertheless, we build upon a long history of manifold identification theory. We give below a partial overview of the concepts that paved our way.

Nonsmooth Optimization

Nonsmoothness naturally appears in various applications of optimization, *e.g.* in decomposition methods in operations research (Briant et al., 2008; de Oliveira et al., 2011), in addition to the examples mentioned above in data analysis.

However, non-differentiability is generally a burden in terms of optimization and is often associated with algorithms based on subgradients (Shor, 2012), inexact proximal

points (Rockafellar, 1976), gradient sampling (Burke et al., 2005), bundle models (Kiwiel, 2006; Oliveira and Sagastizábal, 2014), smoothing (Nesterov, 2005), etc.

In our context, the proximity operator can be computed explicitly (this is S_1), we will thus focus on proximal methods. Furthermore, since large scale applications in data science have given a new youth to first-order optimization, we will particularly focus on proximal gradient methods (see e.g. the recent (Teboulle, 2018)) and coordinate descent algorithms (see e.g. the review (Wright, 2015)).

Notably, we wish to see how these methods numerically benefit from the presence of structure (this is S_2). This means studying the smooth substructure present in these nonsmooth objectives functions. To do so, we first need to take a look at structure identification.

Identification

Let us start with an example. The popular ℓ_1-norm regularization ($g = \|\cdot\|_1$) promotes optimal solutions with few nonzero elements, and its associated proximity operator $\operatorname{prox}_{\gamma g}$ (see Example 2.24) puts the entries with absolute value smaller than γ exactly to 0. Thus, if some algorithm produces a sequence $y_k \to \overline{y}$, then $\operatorname{prox}_{\gamma g}(y_k)$ will have a sparsity pattern corresponding to the smaller coordinates of y_k. Moreover, as (y_k) converges, this pattern will become stable, close to the one $\overline{y}$. This is an example of *identification*.

More generally, the iterates produced by proximal methods eventually reach some low-complexity pattern close to the one of the optimal solution in a finite number of iteration. This active-set identification property is typical of constrained convex optimization (see *e.g.* (Wright, 1993)) and nonsmooth optimization (see *e.g.* (Hare and Lewis, 2004)).

The study of identification dates back at least to (Bertsekas, 1976) who showed that the projected gradient method identifies a sparsity pattern when using non-negative constraints. Such identification has been extensively studied in more general settings (Burke and Moré, 1988; Drusvyatskiy and Lewis, 2014; Lewis, 2002; Lewis and Liang, 2018) (among other references).

Along with structure identification comes the idea of structure stability, *i.e.* how stable is the identified pattern. Coming back to ($\mathcal{P}_A$), a minimizer $\bar{x}$ of this problem satisfies $0 \in \nabla f(\bar{x}) + \partial g(\bar{x}) \Leftrightarrow -\nabla f(\bar{x}) \in \partial g(\bar{x})$. The nonsmoothness of g at $\bar{x}$ determines how much f can be (smoothly) changed while keeping $\bar{x}$ a non-differentiability point of g, which means keeping the identified structure. Such study is part of the general sensitivity analysis of optimization problems; we refer the reader to (Bonnans and Shapiro, 2013; Dontchev, 1983; Mordukhovich, 1992) on that topic.

Harnessing structure

The lower dimensionality of the identified structure and the superior properties of objective function on this subspace can be used numerically. However, this has to be done with care since identification results guarantee that a pattern is identified after a finite *but unknown* time.

In some situations, structure identification can be guaranteed by screening possible patterns (see *e.g.* (Fercoq et al., 2015; Ogawa et al., 2013) for sparse structures). In that case, the problem can be readily restricted to the identified pattern, and the optimization procedure can continue on this lower dimensional problem or update the parameters of the method (Liang et al., 2017a).

We focus here on the general case where we do not have access to the identification time. Hence, in order to harness the structure uncovered along the iterates, adaptivity is key. Exploiting this underlying substructure to develop faster methods has attracted a lot of attention in nonsmooth optimization, pioneered by the developments around the so-called $\mathcal{U}$-Newton method (Lemaréchal et al., 2000) and the notion of partial smoothness (Lewis, 2002). We refer to (Sagastizábal, 2018) for a recent review of these nonsmooth Newton methods relying on an implicit smooth substructure.

However, these methods exploit *inexact* structure information. For instance, the $\mathcal{UV}$-Newton bundle method of (Mifflin and Sagastizábal, 2005) approximates the substructure from first-order information, the recent k-bundle Newton method of (Lewis and Wylie, 2019) refines the approximation from a partial second-order oracle.

In our case, the access to an exact proximity operator (S_1) allows to *exactly* identify the structure. This enables us to take full advantage of this conceptual idea of using higher-order methods using the locally identified structure.

Objectives of the part

First, in Chapter 3, we will investigate the mathematics of proximal identification. In particular, we will exhibit simple conditions for finite time identification, either partial or exact. Then, we will show that together with partial smoothness, exact identification also brings smoothness to the proximity operator itself. These results give overall a rather complete analysis of the structure identification and local smoothness of proximal methods.

Chapter 4 then focuses on the proximal gradient algorithm and investigates the interplay between convergence speed, monotonicity, and identification. The goal of this chapter is to give a practical viewpoint on identification.

However, while we are able to observe the iterates structure along the way, we do not know in general if this structure is optimal/final or even if it will remain stable. The key question is then

How can we harness this identified structure?

As a general idea, once a manifold has been identified (*e.g.* some coordinates are null), there is some hope that it will stay identified for all subsequent iterations (*e.g.* the coordinates will remain null). It is thus natural to update preferentially along this manifold (*e.g.* update preferentially the non-null coordinates). This leads to the following conceptual algorithm:

- Perform a proximity operation $x_k = \operatorname{prox}_{\gamma g}(y_{k-1})$ to identify some structure;
- Observe the current manifold $\mathcal{M}_k \ni x_k$;
- Update preferentially along $\mathcal{M}_k$.

Hence, we present in the following chapters two types of strategies for exploiting the knowledge of the currently identified structure to a numerical advantage without jeopardizing convergence:

- In Chapter 5, we show how to do proximal coordinate descent with an adaptive sampling that favors the directions within the identified manifold.
- In Chapter 6, we investigate how alternating proximal gradient steps and Riemannian Newton steps on the identified manifold can bring super-linear convergence.

3 Proximal Identification & Partial Smoothness

Si l'on n'est plus que mille, eh bien, j'en suis ! Si même
Ils ne sont plus que cent, je brave encor Sylla ;
S'il en demeure dix, je serai le dixième ;
Et s'il n'en reste qu'un, je serai celui-là !
Victor Hugo – *Ultima Verba* in *Les châtiments (1853)*

The purpose of this chapter is to build the theory of proximal identification that will be central in the next chapters. In particular, we review how partial identification can happen with very few hypotheses; this will guide us to investigate structure-adaptive methods in the following. Then, we lay the theoretical grounding to show that optimization methods exhibit a boost in performance if some manifold is exactly identified.

This chapter borrows its key ideas and results from:

- G. Bareilles, F. Iutzeler: On the Interplay between Acceleration and Identification for the Proximal Gradient algorithm , Computational Optimization and Applications, vol. 77, no. 2, pp. 351–378, 2020.
- F. Iutzeler, J. Malick: Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications, to appear in Set-Valued and Variational Analysis, vol. 28, no. 4, pp. 661–678, 2020.
- G. Bareilles, F. Iutzeler, J. Malick: Newton acceleration on manifolds identified by proximal-gradient methods, 2021.

The set of non-differentiability points of a function often has a null Lebesgue measure (this is actually true for all Lipchitz functions by Rademacher's theorem). In fact, in most situations of interest in data science, these non-differentiability points locally have a manifold structure.

Take a manifold $\mathcal{M}$, a sequence (x_k) is said to *identify* manifold $\mathcal{M}$ if

$$x_k \in \mathcal{M} \text{ after some finite time}$$

i.e. for all $k \geq K$ for some finite K. Now, if $\mathcal{M}$ corresponds to non-differentiability points of g, what kind of algorithm can produce a sequence that identifies $\mathcal{M}$? In other words, which algorithms can recover the nonsmooth structure of g?

Intuitively, it is not enough to get x_k close to $\mathcal{M}$, we want a method capable to get x_k exactly in $\mathcal{M}$, which corresponds to having $\partial g(x_k)$ not reduced to a singleton. Since

we are considering a property on the subgradient at the *output* x_k, it seems natural to rely on an implicit subgradient step, *i.e.* a proximity operator.[17]

[17] This is not the only alternative; for instance, if g is polyhedral, a sufficiently rich cutting plane model can find non-differentiability points.

A typical example of this phenomenon is the ℓ_1 norm, for which the proximal operator, also called soft-thresholding, induces sparse outputs (see Example 2.24) which corresponds to the non-differentiability points of the function. This means that an optimization method featuring a soft-thresholding operation can identify sparsity patterns.

Moreover, the non-differentiability points of a nonsmooth function may locally form a smooth manifold on which the function may be smooth. For instance, this is the case for the distance to the ℓ_2-ball, or the nuclear norm. This means that better convergence guarantees and finer algorithms can be derived locally around identified points.

3.1 Proximal Identification

In this part, we will focus on *proximal identification*, that is the identification properties of sequences involving proximal step

$$x_k = \operatorname{prox}_{\gamma g}(y_k) \tag{3.1}$$

and especially converging ones *i.e.* when $y_k \to \overline{y}$.

Nonsmoothness is again of utmost importance. Indeed, in $\mathbb{R}^n$ consider a point $\bar{x}$ belonging to a p-dimensional manifold $\mathcal{M}$ locally defined by the local equation $\Phi(x) = 0$ (with Φ a C^2 mapping, see Section 2.4.1). Take a proper lower-semi continuous function $F : \mathbb{R}^{n-p} \to \overline{\mathbb{R}}$ such that $F(u) \geq 0$ with equality if and only if $u = 0$ and define $g = F \circ \Phi$. Then, locally around $\bar{x}$, $\operatorname{prox}_{\gamma g}$ is well-defined and unique (since g is prox-bounded and prox-regular for $0 \in \partial g(\bar{x})$ with thresholds 0, see Section 2.2), the proximal step (3.1) then gives[18]

[18] The chain rule holds since the Jacobian of the local equation $J\Phi(x_k)$ is of rank $n - p$; see (Rockafellar and Wets, 2009, Ex. 10.7).

$$\frac{y_k - x_k}{\gamma} \in \partial g(x_k) = J\Phi(x_k)^\top \partial F(\Phi(x_k))$$

which means that

- if g is differentiable around x_k (*i.e.* if F is differentiable around $\Phi(x_k)$), then as any point on the manifold $x \in \mathcal{M}$ is a local minimum, $x_k \in \mathcal{M} \Rightarrow \nabla g(x_k) = 0 \Rightarrow x_k = y_k$ which means that the manifold $\mathcal{M}$ is identified only if y_k already belongs to it.
- on the contrary, if g is non-differentiable for all $x \in \mathcal{M}$ near x_k (*i.e.* if F is non-differentiable at 0), $x_k \in \mathcal{M} \Rightarrow y_k - x_k \in \gamma \partial g(x_k) \neq \{0\}$ which means that there is a closed set with non-empty relative interior around x_k in $\mathbb{R}^n$ whose points will be mapped onto the manifold, which is the desired property.

Remark 3.1 (subgradient identification). Identification can also be captured by a converging sequence pair $(x_k, u_k) \to (\overline{x}, \overline{u})$ in terms of subgradients

$$u_k \in \partial g(x_k)$$

as considered in *e.g.* (Fadili et al., 2018; Hare and Lewis, 2004). Nonsmoothness is also necessary (the same example as above can be used to see it). ◀

3.2 Identification under a proximal qualifying condition

Let us focus on the identification properties of proximal algorithms and try to develop the most direct theory possible.

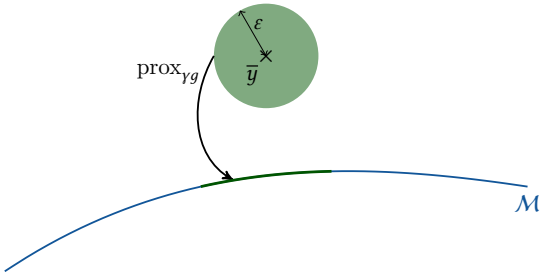
3.2.1 A simple and direct condition for proximal identification

Consider a sequence (y_k) such that $y_k \to \overline{y}$ and define a sequence (x_k) satisfying $x_k \in \operatorname{prox}_{\gamma g}(y_k)$. For identification to happen, *i.e.* to have $x_k \in \mathcal{M}$ after some finite time, a sufficient condition is that for any y close to $\overline{y}$, the proximity operator maps y to $\mathcal{M}$. Mathematically, this *proximal qualifying condition* writes[19]

$$\exists\, \varepsilon > 0 \text{ such that for all } y \in \mathcal{B}(\overline{y}, \varepsilon), \quad \operatorname{prox}_{\gamma g}(y) \in \mathcal{M}. \tag{PQC}$$

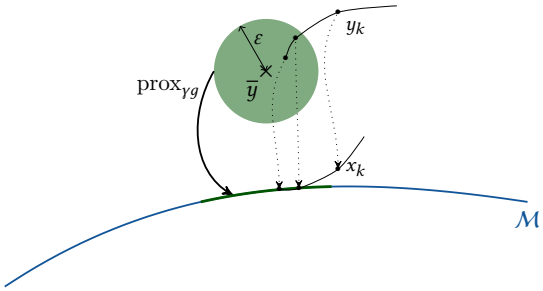
[19] In particular, (PQC) implies that $\operatorname{prox}_{\gamma g}(\overline{y})$ belongs to $\mathcal{M}$. We also make a slight abuse of notation here: if $\operatorname{prox}_{\gamma g}$ is set-valued, then $\operatorname{prox}_{\gamma g}(y) \in \mathcal{M}$ is to be understood as $u \in \mathcal{M}$ for all $u \in \operatorname{prox}_{\gamma g}(y)$.

Notice that this condition only involves the *antecedent* $\overline{y}$ of $\overline{x}$ by the proximity operator. We are interested in the convergence and identification of (x_k) but, in contrast with the literature, the condition here lies on $\overline{y}$ which will be in general different from x. This is because we are considering the proximity operator directly and not only the subgradient of g in order to get a first straightforward result. In the following, we will link this result to standard *subgradient qualifying condition* under additional assumptions in Section 3.3.1 and later provide practical instantiations of this result (specifying $\overline{y}$) in Section 3.4.



Under this condition, it suffices to say that since $y_k \to \overline{y}$, after some finite *but unknown* time, $y_k \in \mathcal{B}(\overline{y}, \varepsilon)$ which directly implies that $x_k \in \operatorname{prox}_{\gamma g}(y_k) \in \mathcal{M}$.

Theorem 3.2 (Identification from (PQC))**.** *Let (x_k) and (y_k) be a pair of sequences such that $x_k \in \operatorname{prox}_{\gamma g}(y_k)$ and $\mathcal{M}$ be a manifold. If $y_k \to \overline{y}$ and (PQC) holds, then, after some finite time, $x_k \in \mathcal{M}$.*



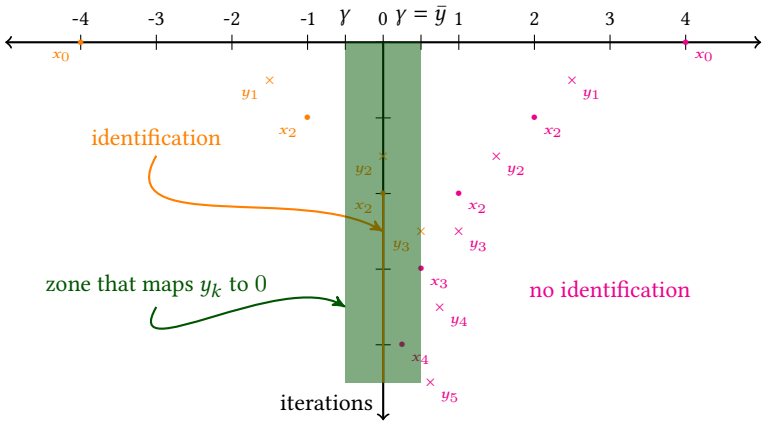


Figure 3.1: Depiction of the situation of Remark 3.3 with $\gamma = 0.5$. The initials points are chosen as $x_0 = -4$ and $x_0 = 4$.

A first comment that directly arises is that the identification time is *unknown*, which is a property generally shared by identification results. Here, it depends on both i) the radius ε of projection onto $\mathcal{M}$ in (PQC) and ii) the time after which the sequence (y_k) belongs to $\mathcal{B}(\overline{y}, \varepsilon)$. The only results in that flavor come from postulating that (PQC) holds for some known $\varepsilon > 0$ and then using the convergence rate of (y_k) (obtained externally by precising and studying the algorithm) to obtain a complexity estimate of the identification time; see *e.g.* (Garrigos et al., 2020; Nutini et al., 2019; Sun et al., 2019).

Remark 3.3 (Failure cases). If one cannot find a ball centered on $\overline{y}$ mapped to $\mathcal{M}$ by $\operatorname{prox}_{\gamma g}$, one can have $x_k \to \overline{x}$ with $\overline{x} \in \mathcal{M}$ but $x_k \notin \mathcal{M}$ for every iteration.

For example, with $f = \frac{1}{2}(\cdot - 1)^2$ and $g = |\cdot|$, the composite problem $\min_{x \in \mathbb{R}} \frac{1}{2}(x-1)^2 + |x|$ has solution $x^\star = 0$ (as the optimality condition writes $0 \in x^\star - 1 + \partial|x^\star|$). The proximal gradient iterates are then $y_k = x_{k-1} - \gamma \nabla f(x_{k-1})$ and $x_k = \operatorname{prox}_{\gamma g}(y_k)$ with $\gamma \in (0, 1)$.

If $x_{k-1} > 0$, $y_k = (1-\gamma)x_{k-1} + \gamma > \gamma$ thus $x_k = y_k - \gamma = (1-\gamma)x_{k-1}$. Hence, for a positive starting point $x_0 > 0$, $x_k = (1-\gamma)^k x_0 \to x^\star = 0$. Therefore, when considering the manifold $\mathcal{M} = \{0\}$, $x^\star$ belongs to $\mathcal{M}$ and yet none of the iterates do. This example illustrates the failure of (PQC). Indeed, $y_k = \gamma + (1-\gamma)^k x_0 \to \overline{y} = \gamma$ but there is no ball around $\overline{y}$ that is mapped to $\mathcal{M} = \{0\}$ since $\operatorname{prox}_{\gamma g}(\overline{y} + \varepsilon) = \varepsilon \notin \mathcal{M}$ for any $\varepsilon > 0$.

If $x_{k-1} < 0$, $y_k = (1-\gamma)x_{k-1} + \gamma < \gamma$ thus $x_k = \min(0, y_k + \gamma)$. Thus, if the starting point is negative ($x_0 < 0$), x_k will increase at each iteration (by at least 2γ) as long as $x_k \leq -2\gamma/(1-\gamma)$ and then $x_k = 0$. This means that if $x_0 < 0$, identification happens in finite time, even though $\overline{y} = \gamma$ as in previous case.

All in all, when $\operatorname{prox}_{\gamma g}(\overline{y})$ belongs to $\mathcal{M}$ but (PQC) does not hold, identification may or may hold depending on the algorithm, initialization, etc. This example is represented by Fig. 3.1. ◀

3.2.2 *Identification with multiples manifolds*

The type of nonsmoothness encountered in machine learning objectives is often linked to user-defined priors which means that it is chosen and relatively simple. However, it

may be impossible to represent the whole structure at scrutiny as a single manifold. In addition, it is often more in line with the target application to define a collection of possible structures, and eventually consider their intersection. For instance, if one seeks sparsity patterns, one can define for each coordinate the manifold corresponding to the nullity of this coordinate, study identification on each of these manifold, and finally intersect them[20].

[20] An alternative, developed in (Fadili et al., 2018), would be to generate a stratification of the whole space, *i.e.* dividing the space into non-overlapping sets, each corresponding to a particular kind of structure, and studying the local properties around some point's strata.

Hence, let us define a finite collection of manifolds:

$$\mathsf{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}.$$

In order to properly study membership in this collection, let us define the *structure* of a point $x \in \mathbb{R}^n$ relatively to the collection C as the mapping $\mathsf{S} : \mathbb{R}^n \to \{0,1\}^q$ defined as

$$\mathsf{S}^{[i]}(x) = 0 \quad \text{if } x \in \mathcal{M}_i \text{ and 1 elsewhere}$$

where $\mathsf{S}^{[i]}(x)$ stands for the i-th coordinate of $\mathsf{S}(x)$.

Using the same reasoning that leads to Theorem 3.2, we can consider the structure of all points near the limit $\overline{y}$, then after some finite time, the iterates (x_k) will at least have the minimal structure common to these neighboring points.

Theorem 3.4 (Partial identification)**.** *Let (x_k) and (y_k) be a pair of sequences such that $x_k \in \operatorname{prox}_{\gamma g}(y_k)$. If $y_k \to \overline{y}$ then, for any $\varepsilon > 0$ and any $i = 1, .., q$, after some finite time*

$$\mathsf{S}^{[i]}(x_k) \;\leq\; \max\left\{\mathsf{S}^{[i]}\big(\operatorname{prox}_{\gamma g}(y)\big) \,:\, \|y - \overline{y}\| \leq \varepsilon\right\}$$

If in addition, $x_k \to \overline{x}$ then, after some finite time $\mathsf{S}(\overline{x}) \;\leq\; \mathsf{S}(x_k)$.[21]

[21] *The inequality is elementwise here.*

Proof. For the first part of the result, since $y_k \to \overline{y}$, we have that for any $\varepsilon > 0$, $\|y_k - \overline{y}\| \leq \varepsilon$ for k large enough. This directly gives the inequality as $\mathsf{S}\big(\operatorname{prox}_{\gamma g}(y_k)\big) \leq \max\left\{\mathsf{S}\big(\operatorname{prox}_{\gamma g}(y)\big) \,:\, \|y - \overline{y}\| \leq \varepsilon\right\}$.

The second part of the result is a bit different. Consider the collection of sets to which $\overline{x}$ belongs $\overline{\mathsf{C}} = \{\mathcal{M}_i \in \mathsf{C} : \overline{x} \in \mathcal{M}_i\}$. Since $\mathcal{M}^\circ := \bigcup\{\mathcal{M} \in \mathsf{C} \setminus \overline{\mathsf{C}}\}$ is a closed set (given that it is a finite union of closed sets), its complementary set $\mathbb{R}^n \setminus \mathcal{M}^\circ$ is open. Since $\overline{x} \in \mathbb{R}^n \setminus \mathcal{M}^\circ$ by definition of $\overline{\mathsf{C}}$, there exists a ball of radius $\varepsilon' > 0$ around $\overline{x}$ that is included in $\mathbb{R}^n \setminus \mathcal{M}^\circ$. Hence, as $x_k \to \overline{x}$, it will reach this ball in finite time thus belong to fewer subspaces than $\overline{x}$ which means that $\mathsf{S}(\overline{x}) \leq \mathsf{S}(x_k)$ coordinate-wise. □

This result thus provides a "sandwich" inequality for converging sequence pairs (x_k) and (y_k) with $x_k \in \operatorname{prox}_{\gamma g}(y_k)$, valid after some finite but unknown time:

$$\mathsf{S}(\overline{x}) \;\leq\; \mathsf{S}(x_k) \;\leq\; \max\left\{\mathsf{S}\big(\operatorname{prox}_{\gamma g}(y)\big) \,:\, \|y - \overline{y}\| \leq \varepsilon\right\} \tag{3.2}$$

where the max is taken over y coordinate-wise. In particular, this result comes with the idea of *partial identification* where only a fraction of the final structure may be recovered. More precisely, a point x_k near $\overline{x}$ have:

- *less structure than $\overline{x}$.* The left-hand inequality guarantees that only the optimal manifolds can be identified, but some might be missed (which means that less structure is identified, no matter which algorithm is used).
- *some structure.* The identified structure is not random: it is controlled by the right-hand-side term, which encompasses how the structure changes around the pair $(\overline{x}, \overline{y})$. This upper-bound is in general impossible to evaluate a priori

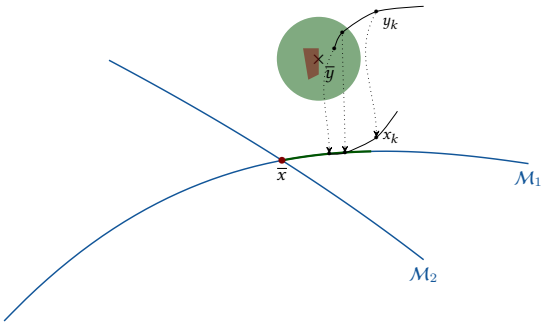


Figure 3.2: Illustration of partial identification. There is a ball (in green) around $\overline{y}$ that maps onto $\mathcal{M}_1$ but not necessarily onto $\mathcal{M}_2$. In this situation, there is a region (in orange) containing $\overline{y}$ that maps onto $\mathcal{M}_2$ but $\overline{y}$ is not in its interior; thus, depending on the sequence (y_k), $\mathcal{M}_2$ may or may not be identified.

(for an exception, see (Duval and Peyré, 2017)). Intuitively, its size captures the difficulty of reaching the structure of $\overline{x}$.

Note that the structure mentioned here is relative to the point towards which the algorithm is converging. If there are multiple solutions to a problem, they may have different sparsity patterns, and thus identification, while still in place may lead to different structures.

Remark 3.5 (How to handle intersections? Direct approach vs. Stratification). Theorem 3.4 and its proof are largely inspired by the reasoning of (Fadili et al., 2018, Th. 1); however, the mathematical objects at play are quite different.

In (Fadili et al., 2018), the authors consider a *stratification* of $\mathbb{R}^n$, *i.e.* a partition $\mathsf{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}$ such that $\mathcal{M} \cap \mathrm{cl}(\mathcal{M}') \neq \emptyset \implies \mathcal{M} \subset \mathrm{cl}(\mathcal{M}')$, in which case $\mathcal{M} \leq \mathcal{M}'$ (which defines a partial ordering). Placing themselves in the convex case, the authors introduce the operators $\mathsf{J}(\mathcal{M}) = \cup_{x \in \mathcal{M}} \mathrm{ri}\, \partial g(x)$ and $\mathsf{J}^*(\mathcal{M}) = \cup_{x \in \mathcal{M}} \mathrm{ri}\, \partial g^{-1}(x)$. The stratification C corresponds to the nonsmooth structure of g if $\mathcal{M} = \mathsf{J}^*(\mathsf{J}(\mathcal{M}))$ and $\mathcal{M} \leq \mathcal{M}' \Leftrightarrow \mathsf{J}(\mathcal{M}) \geq \mathsf{J}(\mathcal{M}')$. Then, if $(x_k, u_k) \to (\overline{x}, \overline{u})$ with $u_k \in \partial g(x_k)$, we have

$$\mathcal{M}(\overline{x}) \leq \mathcal{M}(\overline{x_k}) \leq \mathcal{M}^{**}(\overline{u})$$

where $\mathcal{M}^{**}(\overline{u}) = \mathsf{J}^*(\mathcal{M}^*(\overline{u}))$ with $\mathcal{M}^*(\overline{u}) = \{\mathsf{J}(\mathcal{M}) : \mathcal{M} \in \mathsf{C}, \overline{u} \in \mathsf{J}(\mathcal{M})\}$. We notice again this "sandwich-like" structure, with a right inequality harder to express.

The correspondence between the two families of results will become even clearer in the discussion and figure of Section 3.3.3 which considers the identification of one manifold in the convex case.

On a more personal notice, the results of (Fadili et al., 2018) gave rise to numerous discussions with Jérôme Malick. The results of this subsection are actually an attempt to provide a more practical version of these results by considering directly the proximal operator (replacing J and the subgradient sequences, see also Remark 3.1) and directly considering the proximity operator outputs instead of a stratification of the space (see Section 3.2.3). In that respect, (Fadili et al., 2018) is an intermediate link between the results presented in this section and the more stringent and classical results of identification under partial smoothness presented in Section 3.3.1. ◂

3.2.3 Examples and Structure scrutiny

Example 3.6 (Sparsity). Sparsity patterns can be described using the following collection of manifolds:

$$\mathcal{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \quad \text{with} \quad \mathcal{M}_i = \{x \in \mathbb{R}^n : x_i = 0\}.$$

Sparse vectors are typically obtained after application of the proximity operator of the ℓ_1-norm (see Example 2.24):

$$\operatorname{prox}^{[i]}_{\gamma\|\cdot\|_1}(y) = \begin{cases} y^{[i]} + \gamma & \text{if } y^{[i]} < -\gamma \\ 0 & \text{if } -\gamma \leq y^{[i]} \leq \gamma \\ y^{[i]} - \gamma & \text{if } y^{[i]} > \gamma \end{cases}.$$

We notice that the structure of $x = \operatorname{prox}_{\gamma\|\cdot\|_1}$ is obtained explicitly during the computation. Indeed, $x \in \mathcal{M}_i$ if and only if $-\gamma \leq y^{[i]} \leq \gamma$. ◀

The observation that structure is explicitly obtained during the proximity operator computation is actually ubiquitous for regularizers in data science problems. Most results developed in the upcoming Chapters 4–6 will be based on the fact that

$$\textit{upon computation of } x = \operatorname{prox}_{\gamma g}(y), \textit{ we naturally obtain } \mathcal{S}(x)$$

for a large class of pertinent functions. For instance, the following examples fall into that category.

Example 3.7 (Piecewise constant). The structure of piecewise constant vectors can be described using the following collection:

$$\mathcal{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_{n-1}\} \quad \text{with} \quad \mathcal{M}_i = \left\{x \in \mathbb{R}^n : x^{[i]} = x^{[i+1]}\right\}.$$

Such vectors can be obtained as the output of the proximity operator of the 1D total variation (see Example 2.27). When computing this proximity operator (see *e.g.* (Condat, 2013)), we obtain as a byproduct of the equivalent dynamic program the structure of the signal. ◀

Example 3.8 (low rank). To express the rank of a matrix $X \in \mathbb{R}^{n_1 \times n_2}$, it is natural to examine its singular values $(\sigma_1, \dots, \sigma_q)$ (where $q = \min\{n_1, n_2\}$). A direct way to promote low-rank matrices is through nuclear norm regularization, which is the ℓ_1-norm of the singular values vector: $g(X) = \|X\|_* = \sum_{i=1}^{q} \sigma_i$, see Example 2.28 for the computation of the proximity operator. Naturally, the collection writes

$$\mathcal{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\} \quad \text{with} \quad \mathcal{M}_i = \{\operatorname{rank}(X) = i\}$$

and the rank of the output is naturally known since the thresholding of the singular values gave the exact number of non-null entries, hence the rank. ◀

3.2.4 Exact Identification

The partial identification result of Theorem 3.4 can be strengthened to an exact version guaranteeing $\mathcal{S}(x_k) = \mathcal{S}(\overline{x})$ after some time. To do so, a direct way mirroring the proximal qualification condition (PQC) is to assume that both "buns" of the sandwich

inequality in Theorem 3.4 are equal. This leads to the following *proximal qualifying condition for multiple manifolds*:

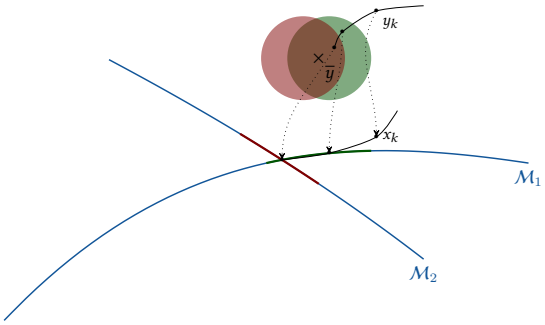
$$\exists\, \varepsilon > 0 \text{ such that for all } y \in \mathcal{B}(\overline{y}, \varepsilon) \quad \mathrm{S}(\mathrm{prox}_{\gamma g}(\overline{y})) = \mathrm{S}(\mathrm{prox}_{\gamma g}(y)) \qquad \text{(PQC-M)}$$

Note that we can see from the proof of Theorem 3.4 that $\mathrm{S}(\mathrm{prox}_{\gamma g}(\overline{y})) \leq \mathrm{S}(\mathrm{prox}_{\gamma g}(y))$ holds on a sufficiently small neighborhood and thus the condition only enforces a one-sided inequality. With this qualification condition, we have the following result.

Corollary 3.9 (Exact identification)**.** *Let (x_k) and (y_k) be a pair of sequences such that $x_k \in \mathrm{prox}_{\gamma g}(y_k)$. If $x_k \to \overline{x}$, $y_k \to \overline{y}$, and (PQC-M) holds, then, after some finite time*

$$\mathrm{S}(x_k) = \mathrm{S}(\overline{x}).$$

Intuitively, getting back to our drawings, the qualification condition implies that the two attractions balls for $\mathcal{M}_1$ and $\mathcal{M}_2$ intersect and $\overline{y}$ lies in the interior of the intersection. Then, the exact identification holds as depicted in the following illustration.



3.2.5 *Identification for stochastic proximal algorithms*

Some proximal methods lead to stochastic processes for their iterates, either coming from some randomized "coordinate descent" (see *e.g.* (Bianchi et al., 2016; Combettes and Pesquet, 2015; Iutzeler et al., 2013a; Richtárik and Takáč, 2014)) or from a stochastic oracle, typically a noisy gradient coming from data sampling (see *e.g.* (Defazio et al., 2014a; Duchi and Singer, 2009; Mairal, 2015)).

In order to benefit from identification with probability 1, stochastic proximal algorithms have to converge almost surely. This is to ensure that for each event, there is a finite time after which the iterates belong to a neighboring ball which leads to identification. This is the case for most randomized and variance reduced methods (Bianchi et al., 2016; Combettes and Pesquet, 2015; Defazio et al., 2014a; Iutzeler et al., 2013a; Mairal, 2015). However, when convergence only holds in probability (*e.g.* for the proximal stochastic gradient (Duchi and Singer, 2009)), the iterates may not recover the structure of the problem as detailed in (Poon et al., 2018, Sec. 1.3).

3.3 Partial smoothness

A key element to study in more details the relationship between proximal points and smooth manifolds is the notion of *partial smoothness* (Lewis, 2002, Def. 2.7). Broadly speaking, a function g is partly smooth with respect to a smooth manifold $\mathcal{M}$ at a point $\bar{x} \in \mathcal{M}$ if, near $\bar{x}$, it is smooth along the manifold $\mathcal{M}$ and nonsmooth across $\mathcal{M}$.

Definition 3.10 (Partial smoothness)**.** A function g is (C^2-)*partly smooth* at a point $\bar{x}$ relative to the C^2 manifold $\mathcal{M}$ around $\bar{x}$ if:

- (smoothness) the restriction of g to $\mathcal{M}$ is a C^2 function near $\bar{x}$;
- (regularity) g is (Clarke) regular at all points $x \in \mathcal{M}$ near $\bar{x}$, with $\partial g(x) \neq \emptyset$;
- (sharpness) the affine span of $\partial g(\bar{x})$ is a translate of $N_{\bar{x}}\mathcal{M}$;
- (sub-continuity) the set-valued mapping ∂g restricted to $\mathcal{M}$ is continuous at $\bar{x}$.

This definition may seem a bit stringent, notably on the smoothness side, but it will actually bring us much more control in terms of continuity around $\bar{x}$ (see Section 3.4). For now, let us simply notice that the sharpness condition was informally evoked when discussing the necessity of non-smoothness in Section 3.1 and the smoothness on the manifold ensures us that there are no other non-differentiability points that could trap proximal points in the neighborhood. This notion is illustrated in Fig. 3.3.

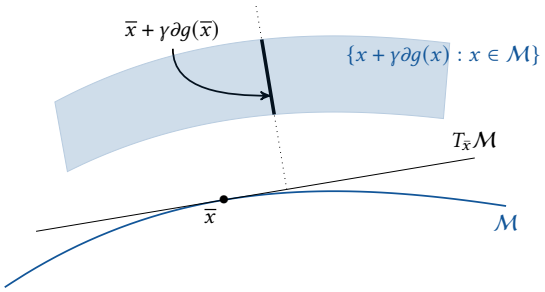


Figure 3.3: Illustration of partial smoothness.

3.3.1 Proximal identification under partial smoothness

Partial smoothness characterizes the function locally, but the point itself has to fulfill some non-degeneracy condition as before. This is most commonly done though a *subgradient qualification condition*, for a proximal method verifying $(x_k, y_k) \to (\bar{x}, \bar{y})$ with $x_k \in \operatorname{prox}_{\gamma g}(y_k)$, this gives:[22]

$$\frac{\bar{y} - \bar{x}}{\gamma} \in \operatorname{ri} \partial g(\bar{x}). \tag{SQC}$$

[22] Even though γ, the stepsize for the proximal operator used is present in the equation, it will in general disappear as soon as the algorithm is made precise, see Section 3.4

Using this condition, we can show that the *proximal* qualification condition holds.

Theorem 3.11. *Suppose that g is both r-prox-regular at $\bar{x}$ and partly-smooth relative to manifold $\mathcal{M}$ at $\bar{x}$. Take $\gamma, \bar{\gamma}$ such that $0 < \gamma < \bar{\gamma} \leq 1/r$ and $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$. If either 1) γ is sufficiently close to $\bar{\gamma}$, or 2) $\bar{y}$ is sufficiently close to $\bar{x}$, then* (SQC) *implies* (PQC)*.*

To be more explicit, (SQC) *implies* (PQC) means that provided that the qualification condition $\frac{\bar{y}-\bar{x}}{\gamma} \in \operatorname{ri} \partial g(\bar{x})$ holds, then $\operatorname{prox}_{\gamma g}(y) \in \mathcal{M}$ for all y sufficiently close to $\bar{y}$.

Proof. Adopting the same reasoning as in (Lewis, 2002, Sec. 5) and (Daniilidis et al., 2006, Sec. 4.1), we consider the function

$$\begin{aligned} \rho : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R}^n \\ (u, y) &\mapsto g(u) + \tfrac{1}{2\gamma}\|u - y\|^2, \end{aligned}$$

and denote by $\rho_y = \rho(\cdot, y)$. Computing the proximal gradient $\mathrm{prox}_{\gamma g}(y)$ can then be seen as minimizing the parametrized function ρ_y as noticed in Section 2.2.

Step 1. As a first step, we study the minimizers of ρ_y restricted to $\mathcal{M}$, for y near $\bar{y}$ and show that they are actually strong minimizers, and evolve near $\bar{x}$ as a continuously differentiable mapping of the parameter y. We consider the parametric manifold optimization problem, for y near $\bar{y}$:

$$\min_{x \in \mathcal{M}} \rho_y(x). \tag{$P_{\mathcal{M}}(y)$}$$

Since g is C^2-partly-smooth relative to $\mathcal{M}$, ρ_y is twice continuously differentiable on $\mathcal{M}$. The prox-regularity and $\gamma < 1/r$ implies that $\bar{x}$ is a strong local minimizer of $\rho_{\bar{y}}$ (see Lemma 2.21), which in turn implies the following second-order optimality condition on $\mathcal{M}$ (see Lemma 2.43):

$$\mathrm{grad}\ \rho_{\bar{y}}(\bar{x}) = 0 \qquad \mathrm{Hess}\ \rho_{\bar{y}}(\bar{x}) \geq \left(\frac{1}{\gamma} - r\right) I \succ 0.$$

We consider the equation $G(x, y) = 0$, for x, y near $\bar{x}, \bar{y}$, where $G : \mathcal{M} \times \mathbb{R}^n \to T\mathcal{B}$ is defined as $G(x, y) = \mathrm{grad}\ \rho_y(x)$. This function is continuously differentiable on a neighborhood of $(\bar{x}, \bar{y})$, and its differential relative to $\bar{x}$ at that point, $\mathrm{Hess}\ \rho_{\bar{y}}(\bar{x})$, is invertible. The implicit function theorem[23] thus grants the existence of neighborhoods $\mathcal{N}_{\bar{x}}$, $\mathcal{N}_{\bar{y}}$ of $\bar{x}$, $\bar{y}$ in $\mathcal{M}, \mathbb{R}^n$, and a continuously differentiable function $\mathrm{p} : \mathcal{N}_{\bar{y}} \to \mathcal{N}_{\bar{x}}$ such that, for any y in $\mathcal{N}_{\bar{y}}$, $G(\mathrm{p}(y), y) = \mathrm{grad}\, \rho_y(\mathrm{p}(y)) = 0$.

[23] See *e.g.* (Halkin, 1974, Th. B).

Step 2. As a second step, we turn to show that the minimizer $\mathrm{p}(y)$ of $\rho_y|_{\mathcal{M}}$ is actually a critical point of ρ_y. More precisely, we claim that, for y near $\bar{y}$ there holds $0 \in \mathrm{ri}\, \partial \rho_y(\mathrm{p}(y))$, that is

$$\frac{1}{\gamma}(y - \mathrm{p}(y)) \in \mathrm{ri}\, \partial g(\mathrm{p}(y)).$$

This property holds at $(\bar{x} = \mathrm{p}(\bar{y}), \bar{y})$ by assumption. By contradiction, assume there exist sequences of points (y_r) with limit $\bar{y}$, $(x_r) = (\mathrm{p}(y_r))$ with limit $\bar{x} = \mathrm{p}(\bar{y})$ and (h_r) of unit norm $\|h_r\| = 1$ such that for all r, h_r separates 0 from $\partial \rho_{y_r}(x_r)$:

$$\inf_{h \in \partial \rho_{y_r}(x_r)} \langle h_r, h \rangle \geq 0.$$

Since (h_r) is bounded, a converging subsequence can be extracted from it, let $\bar{h}$ denote its limit. At the cost of renaming iterates, we assume that $\lim_{r \to \infty} h_r = \bar{h}$. The above property still holds at the limit $r \to \infty$. Indeed, let $\bar{u} \in \partial \rho_{\bar{y}}(\bar{x})$. Since g is partly smooth, the mapping $(x, y) \in \mathcal{N}_{\bar{x}} \times \mathcal{N}_{\bar{y}} \mapsto \partial \rho_y(x) = \partial g(x) + \frac{1}{\gamma}(x - y)$ is continuous. Therefore, there exists a sequence (u_r) such that $u_r \in \partial \rho_{y_r}(x_r)$ and $\lim_{r \to \infty} u_r = \bar{u}$. We have for all r: $\langle u_r, h_r \rangle \geq 0$, which yields at the limit $\langle \bar{u}, \bar{h} \rangle \geq 0$. Thus $\bar{h}$ separates 0 from $\partial \rho_{\bar{y}}(\bar{x})$, which contradicts our assumption that $0 \in \mathrm{ri}\, \partial \rho_{\bar{y}}(\mathrm{p}(\bar{y}))$.

Conclusion. We thus have a continuously differentiable function p defined on a neighborhood of $\bar{y}$ with values on $\mathcal{M}$ such that $0 \in \mathrm{ri}\, \partial \rho_y(\mathrm{p}(y))$.

Thus, we have that $(y - \mathrm{p}(y))/\gamma \in \partial g(\mathrm{p}(y))$. The characterization of proximity by the optimality condition (Theorem 2.18) finally gives that $\mathrm{p}(y) = \mathrm{prox}_{\gamma g}(y)$ for y close enough to $\bar{y}$.

Therefore, we have that the existence of a neighborhood of $\bar{y}$ on which $\mathrm{prox}_{\gamma g}(y) \in \mathcal{M}$ which is exactly (PQC). □

A drawback of this result is its hypotheses on $\gamma, \bar{x}, \bar{y}$ that are due to the need to go from a point $x \in \mathcal{M}$ satisfying $0 \in \rho_y(x)$ to the fact that $x = \operatorname{prox}_{\gamma g}(y)$. These assumptions can be dropped in the convex case (see Section 3.3.3) or by paying a prize on the admissible stepsizes (see the remark below and Section 2.2). Additionally, we also bring the attention of the reader to the fact that the theorem can be readily applied near qualified critical points *i.e.* when $0 \in \operatorname{ri} \partial g(\bar{x})$, as detailed in Section 3.4.

Remark 3.12 (Additional assumptions and stepsize). The assumptions here are a bit stronger than the ones in (Daniilidis et al., 2006). This is because we guarantee here the result for a broad range of stepsizes, not for a sufficiently small one. Adding the assumption that g is prox-bounded, the result can be directly modified to "Take γ sufficiently small, then (SQC) implies (PQC)". This mirrors the difference between Theorem 2.18 and Theorem 2.17. ◀

Now, that we have established the link with the proximal qualification condition, we can readily apply Theorem 3.2 to get the following identification result.

Corollary 3.13 (Identification with partial smoothness). *Let $(x_k) \to \bar{x}$ and $(y_k) \to \bar{y}$ be a pair of converging sequences such that $x_k \in \operatorname{prox}_{\gamma g}(y_k)$ and $\mathcal{M}$ be a manifold. Suppose that g is both r-prox-regular at $\bar{x}$ and partly-smooth relative to $\mathcal{M}$ at $\bar{x}$. Take $\gamma, \bar{\gamma}$ such that $0 < \gamma < \bar{\gamma} \leq 1/r$ and $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$. If*

i) (SQC) *is verified,* i.e. $\frac{\bar{y}-\bar{x}}{\gamma} \in \operatorname{ri} \partial g(\bar{x})$;

ii) either 1) γ is sufficiently close to $\bar{\gamma}$, or 2) $\bar{y}$ is sufficiently close to $\bar{x}$;

then, after some finite time

$$x_k \in \mathcal{M}.$$

3.3.2 Uniqueness of identified manifold

One of the counterparts of assuming smoothness locally on some manifold is the implicit acknowledgement that $\bar{x}$ only belongs to one identified manifold. Thus, no *partial identification* in the sense of the "sandwich inequality" (3.2) can be obtained.[24]

This statement can be formalized as the following proposition, whose proof follows the reasoning of (Hare and Lewis, 2004, Cor. 3.2).

[24] This was one of the motivations behind the notion of mirror stratifiability in (Fadili et al., 2018).

Proposition 3.14 (Uniqueness of manifold). *Consider a function g, two manifolds $\mathcal{M}_1$, $\mathcal{M}_2$ and a point $\bar{x} \in \mathcal{M}_1 \cap \mathcal{M}_2$ such that g is prox-bounded, r-prox-regular at $\bar{x}$, and partly-smooth relative to both manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$. Then, near $\bar{x}$, $\mathcal{M}_1 = \mathcal{M}_2$.*

Proof. For the sake of eventual contradiction, let (x_k) denote any sequence converging to $\bar{x}$ such that $x_k \in \mathcal{M}_1 \setminus \mathcal{M}_2$ for all k. Since g is prox-regular, (Rockafellar and Wets, 2009, Prop. 13.37) tells us that there is $\bar{\gamma} > 0$ such that $\bar{x} = \operatorname{prox}_{\bar{\gamma} g}(\bar{y})$ for some $\bar{y} \in \bar{x} + \bar{\gamma} \operatorname{ri} \partial g(\bar{x}) \in \mathbb{R}^n$ (since g is partly smooth, $\partial g(\bar{x})$ has non-empty relative interior, and $\bar{y}$ can be taken as $\bar{x} + \bar{\gamma}\bar{v}$ for any $\bar{v} \in \operatorname{ri} \partial g(\bar{x})$ by reasoning as in the proof of (Hare and Sagastizábal, 2009, Th. 4)). As g is partly smooth relative to $\mathcal{M}_1$, we know that ∂g is outer semi-continuous (Rockafellar and Wets, 2009, Prop. 8.7). Then, any converging sequence of subdifferentials $v_k \in \partial g(x_k)$ converges to a point in $\partial g(\bar{x})$.

We can thus select a sequence $v_k \in \partial g(x_k)$ converging to $\bar{v} = (\bar{y} - \bar{x})/\bar{\gamma} \in \operatorname{ri} \partial g(\bar{x})$ and define $y_k = x_k + \gamma v_k$ for some $\gamma \in (0, \bar{\gamma})$. It is immediate to see that the sequence (y_k) converges to $y^\gamma = (1 - (\gamma/\bar{\gamma}))\bar{x} + (\gamma/\bar{\gamma})\bar{y}$ and that y^γ can be made arbitrarily close to $\bar{y}$ by taking γ close to $\bar{\gamma}$. Thus, we can consider that, properly choosing γ, y_k reaches any neighborhood of $\bar{y}$ in a finite number of iterations.

Theorem 2.18 then indicates that for k large enough, we have $x_k = \operatorname{prox}_{\gamma g}(y_k)$. Furthermore, Corollary 3.13 applied with $\mathcal{M} = \mathcal{M}_2$ shows that $\operatorname{prox}_{\gamma g}$ is $\mathcal{M}_2$-valued near $\bar{y}$ which implies that $x_k = \operatorname{prox}_{\gamma g}(y_k) \in \mathcal{M}_2$ for k large enough which contradicts x_k being in $\mathcal{M}_1 \setminus \mathcal{M}_2$. □

Note that manifolds can be included one in another, for instance if $\mathcal{M}_1$ is the manifold of rank-1 matrices and $\mathcal{M}_2$ is the manifold of rank-2 matrices. For these situations, we will always consider the smallest manifold in practice as guided by theory which says that if g is partly smooth with respect to $\mathcal{M}_1$, it cannot be partly smooth at the same point with respect to $\mathcal{M}_2$ because of the absence of sharpness.

3.3.3 *The relation between (PQC) and (SQC) in the convex case*

In the convex case, the notion of prox-regularity can be dropped and the result simplified since $(y-x)/\gamma \in \partial g(x)$ directly implies that $x = \operatorname{prox}_{\gamma g}(y)$ (Theorem 2.22 suffices, there is no need to call Theorem 2.18 which is the source of the assumptions on γ and $\bar{y}$). Then, Corollary 3.13 can be simplified as follows.

Corollary 3.15 (Identification with partial smoothness: convex case)**.** *Let $(x_k) \to \bar{x}$ and $(y_k) \to \bar{y}$ be a pair of converging sequences such that $x_k = \operatorname{prox}_{\gamma g}(y_k)$ and $\mathcal{M}$ be a manifold. Suppose that g is convex, lower semi-continuous, and partly-smooth relative to $\mathcal{M}$ at $\bar{x}$. If (SQC) is verified,* i.e. *if $\frac{\bar{y}-\bar{x}}{\gamma} \in \operatorname{ri} \partial g(\bar{x})$, then after some finite time*

$$x_k \in \mathcal{M}.$$

Proof. The proof of Theorem 3.11 can be directly followed with a prox-regularity threshold $r = 0$ since we are in the convex case. The only changes lies in the conclusion where $(y - \mathrm{p}(y))/\gamma \in \partial g(\mathrm{p}(y))$ directly implies that $\mathrm{p}(y) = \operatorname{prox}_{\gamma g}(y)$ by Theorem 2.22. Then, Theorem 3.2 can be applied to get the result. □

In fact, convexity, or, more precisely, the equivalence between $(y - x)/\gamma \in \partial g(x)$ and $x = \operatorname{prox}_{\gamma g}(y)$ enables to get a better intuition about the relation between the proximal qualifying condition (PQC) and the subgradient qualifying condition (SQC).

From (SQC) to (PQC): (SQC) means that $\bar{y} \in \operatorname{ri}(\bar{x}+\gamma\partial g(\bar{x}))$ so there is an open set in the affine hull of $\bar{x}+\gamma\partial g(\bar{x})$ (represented as the dotted line in Fig. 3.4) which contains $\bar{y}$. Thus, there is some $\varepsilon > 0$ such that all $y \in \mathcal{B}(\bar{y}, \varepsilon) \cap \{\bar{x} + \gamma\partial g(\bar{x})\}$, $\operatorname{prox}_{\gamma g}(y) = \bar{x} \in \mathcal{M}$.

Now, by partial smoothness, the set $\{x+\gamma\partial g(x) : x \in \mathcal{M}\}$ evolves smoothly around $\bar{x} + \gamma\partial g(\bar{x})$. This means that there is an open neighborhood of $\bar{y}$ in $\mathbb{R}^n$ that belongs to $\{x + \gamma\partial g(x) : x \in \mathcal{N}_{\bar{x}} \subset \mathcal{M}\}$. This implies the existence of some $\varepsilon > 0$ such that all $y \in \mathcal{B}(\bar{y}, \varepsilon)$, $y = x + \gamma\partial g(x)$ for some $x \in \mathcal{M}$; or equivalently $\operatorname{prox}_{\gamma g}(y) = x \in \mathcal{M}$, which is exactly (PQC).

From (PQC) to (SQC): (PQC) means that there is a ball of radius ε around $\bar{y}$ such that every y in that ball verifies $\operatorname{prox}_{\gamma g}(y) \in \mathcal{M}$. In particular, $\bar{x} := \operatorname{prox}_{\gamma g}(\bar{y}) \in \mathcal{M}$.

Using partial smoothness, $\partial g(\bar{x})$ has its affine hull orthogonal to the tangent space of $\mathcal{M}$ at $\bar{x}$. If we intersect the affine hull of $\bar{x} + \gamma\partial g(\bar{x})$ and the ball around $\bar{y}$, we notice that every point in the intersection is mapped to $\bar{x}$ by $\operatorname{prox}_{\gamma g}$ hence the intersection is included in $\bar{x} + \gamma\partial g(\bar{x})$. All in all, this means that $\bar{y}$ belongs to an open set within the affine hull of $\bar{x} + \gamma\partial g(\bar{x})$, *i.e.* $\bar{y} \in \operatorname{ri}(\bar{x} + \gamma\partial g(\bar{x}))$, which is exactly (SQC).

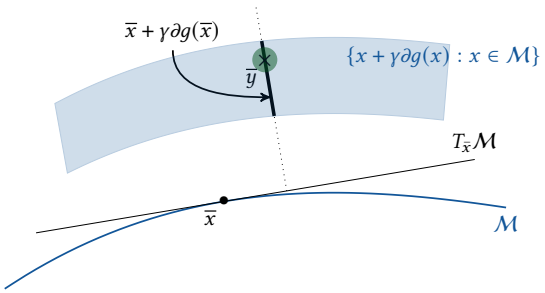


Figure 3.4: Relation between (PQC) and (SQC) under partial smoothness in the convex case.

3.4 Local smoothness around critical points

The previous sections introduced the main tools for proximal identification using first the *proximal qualification condition* (Section 3.2) and then using partial smoothness (Section 3.3). However, the last results (especially Theorem 3.11) tend to show that partial smoothness is more delicate to use since it requires considering only one manifold and satisfying conditions on the proximity operator (existence of a proximity operator with a greater stepsize, subgradient qualification, etc.) that make this notion best used around fixed points of the proximity operator. In that situation, partial smoothness has an advantage over proximal qualification: it gives much finer results in terms of smoothness of the proximal operation.

Indeed, the proof of Theorem 3.11 is based on the study of $\rho_y(x) = g(x) + \frac{1}{2\gamma}\|x-y\|^2$ and is comprised of two parts:

- proving the existence of critical points on the manifold $\mathcal{M}$ that are continuous with respect to y;
- showing that the critical points of ρ_y restricted to the manifold $\mathcal{M}$ are critical points of ρ_y on the whole space (and thus proximal points), using subgradient qualification (SQC) and the continuity of ∂g brought by partial smoothness.

The first step is done by considering the equation $G(x, y) = \operatorname{grad} \rho_y(x) = 0$ in a neighborhood of $(\bar{x}, \bar{y})$ in $\mathcal{M} \times \mathbb{R}^n$ and applying the implicit function theorem to get the existence of a function $\mathrm{p} : \mathbb{R}^n \to \mathcal{M}$ such that $G(\mathrm{p}(y), y) = \operatorname{grad} \rho_y(\mathrm{p}(y)) = 0$. The implicit function further tells us that $\mathrm{p}(y)$ (that is shown to be exactly $\operatorname{prox}_{\gamma g}(y)$ in the second step) is actually C^1 since g is C^2 from the definition of partial smoothness.

Thus, partial smoothness enables to show the smoothness of the proximal mapping, which is important in terms of stability of the associated algorithms (in particular, this will be crucial in Chapter 6). In order to better illustrate this property, we will particularize our results by specifying the value of the antecedent $\bar{y}$ for common proximal methods. Note that the results presented here are local and the identification guarantees additionally assume the convergence of the method.

3.4.1 For the proximal point

As presented in Section 2.3.2, the Proximal point algorithm

$$x_{k+1} = \operatorname{prox}_{\gamma g}(x_k) \qquad \text{(Proximal point)}$$

aims at finding a minimizer of g by finding a fixed point of the associated proximity operator. By taking $\bar{y} = \bar{x}$ in the result of Section 3.3.1 (out of simplicity and without loss of generality, we also take $\bar{\gamma} = 1/r$). We note that the subgradient qualification condition (SQC) now boils down to the more common critical point qualification condition

$$0 \in \operatorname{ri} \partial g(\bar{x}). \tag{QC}$$

Combining Theorem 3.11 and Corollary 3.13 directly gives the following result.

Theorem 3.16. *Let $\bar{x}$ be a point of a manifold $\mathcal{M}$ and r such that:*

i) (qualification condition) $0 \in \operatorname{ri} \partial g(\bar{x})$;

ii) (proximal stability) $\bar{x} = \operatorname{prox}_{g/r}(\bar{x})$;

iii) (prox-regularity) g is r prox-regular at $\bar{x}$;

iv) (partial smoothness) g is partly smooth at $\bar{x}$ with respect to $\mathcal{M}$.

Then, for any $\gamma \in (0, 1/r)$,

a) (local smoothness) the proximity operator $\operatorname{prox}_{\gamma g}$ is C^1 and $\mathcal{M}$ valued near $\bar{x}$;

b) (identification) if $y_k \to \bar{x}$, then $\operatorname{prox}_{\gamma g}(y_k) \in \mathcal{M}$ for k large enough.

This type of result and assumptions is common in the literature; see *e.g.* (Daniilidis et al., 2006, Th. 28). However, the proximal stability condition is sometimes eluded (it is naturally verified in the convex case), or replaced by prox-boundedness (in which case the stepsize γ has to be sufficiently small).

The interest of Theorem 3.16 for the Proximal point algorithm is that it guarantees that the iterates (x_k) identify the manifold $\mathcal{M}$ and furthermore behave smoothly on it.

3.4.2 For the proximal gradient

As detailed in Section 2.3.3, the Proximal gradient algorithm then consists in iterating

$$x_{k+1} = \operatorname{prox}_{\gamma g}\left(x_k - \gamma \nabla f(x_k)\right) \tag{Proximal gradient}$$

to find a minimizer of $f + g$. Taking $\bar{y} = \bar{x} - \gamma \nabla f(\bar{x})$ in the result of Section 3.3.1, we can quickly show a result similar to Theorem 3.16 with the qualification condition

$$-\nabla f(\bar{x}) \in \operatorname{ri} \partial g(\bar{x}). \tag{QC}$$

More precisely, we get the following result.

Theorem 3.17. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a C^1 function and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ a lower semi-continuous function.*
Let $\bar{x}$ be a point of a manifold $\mathcal{M}$ and r such that:

i) (qualification condition) $-\nabla f(\bar{x}) \in \operatorname{ri} \partial g(\bar{x})$;

ii) (proximal gradient stability) $\bar{x} = \operatorname{prox}_{g/r}(\bar{x} - 1/r \nabla f(\bar{x}))$;

iii) (prox-regularity) g is r prox-regular at $\bar{x}$;

iv) (partial smoothness) g is partly smooth at $\bar{x}$ with respect to $\mathcal{M}$.

Then, for any $\gamma \in (0, 1/r)$,

a) (local smoothness) the proximity operator $\operatorname{prox}_{\gamma g}$ is C^1 and $\mathcal{M}$ valued near $\bar{x} - \gamma \nabla f(\bar{x})$;

b) (identification) if $y_k \to \bar{x} - \gamma \nabla f(\bar{x})$, then $\operatorname{prox}_{\gamma g}(y_k) \in \mathcal{M}$ for k large enough.

This results provides an identification result for the Proximal gradient method. However, it only provides smoothness for the proximity operator, not the full proximal *gradient* operator. This can be corrected rather easily, provided that f is C^2.[25]

[25] Actually, this reasoning can be directly extended to compositions of the proximity operator with a C^1 mapping.

3.4.3 For the proximal gradient, second version

Adopting the same reasoning as in the proof of Theorem 3.11 but incorporating the gradient of f in the surrogate function we obtain the following result.

Theorem 3.18. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a C^2 function and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ a lower semi-continuous function.*
Let $\bar{x}$ be a point of a manifold $\mathcal{M}$ and r such that:

i) (qualification condition) $-\nabla f(\bar{x}) \in \operatorname{ri} \partial g(\bar{x})$;

ii) (proximal gradient stability) $\bar{x} = \operatorname{prox}_{g/r}(\bar{x} - 1/r\nabla f(\bar{x}))$;

iii) (prox-regularity) g is r prox-regular at $\bar{x}$;

iv) (partial smoothness) g is partly smooth at $\bar{x}$ with respect to $\mathcal{M}$.

Then, for any $\gamma \in (0, 1/r)$,

a) (local smoothness) the proximal gradient operator $\operatorname{prox}_{\gamma g}(\cdot - \gamma\nabla f(\cdot))$ is C^1 and $\mathcal{M}$ valued near $\bar{x}$;

b) (identification) if $y_k \to \bar{x}$, then $\operatorname{prox}_{\gamma g}(y_k - \gamma\nabla f(y_k)) \in \mathcal{M}$ for k large enough.

Proof. The main difference with respect to the proof of Theorem 3.11 is that we now consider the function

$$\begin{aligned} \rho : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R}^n \\ (u, y) &\mapsto g(u) + \tfrac{1}{2\gamma}\|u - y + \gamma\nabla f(y)\|^2. \end{aligned}$$

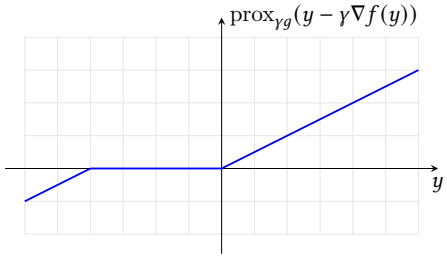
Computing the proximal gradient $\operatorname{prox}_{\gamma g}(y - \gamma\nabla f(y))$ can then be seen as minimizing the parametrized function $\rho_y := \rho(\cdot, y)$. □

The identification properties of the proximal gradient were extensively studied in the literature; *e.g.* (Garrigos et al., 2020; Liang et al., 2017a; Sun et al., 2019). However, the non-convex case is more rarely treated, as well as the local smoothness of the proximal gradient operator.

Remark 3.19 ((Non) smoothness of proximal gradient at non qualified points.). Take the same functions as in Remark 3.3, $f(x) = \frac{1}{2}(x-1)^2$ and $g(x) = |x|$ for any $x \in \mathbb{R}$, and $\gamma \in (0, 1)$. The unique minimizer lies at the origin, but is not a structured critical point as it is not qualified: $0 \notin \operatorname{ri} \partial(f + g)(0) = \operatorname{ri}[-2; 0] = (-2; 0)$. The proximal gradient operator is

$$\operatorname{prox}_{\gamma g}(y - \gamma\nabla f(y)) = \begin{cases} (1-\gamma)y + 2\gamma & \text{if } y \le \frac{-2\gamma}{1-\gamma} \\ 0 & \text{if } \frac{-2\gamma}{1-\gamma} \le y \le 0 \\ (1-\gamma)y & \text{if } 0 \le y \end{cases}$$

and is indeed not continuously differentiable near 0 as illustrated next (for $\gamma = 0.5$).



In this illustration, we see again two phenomena observed in Remark 3.3. First, identification is still possible when coming "from the left", *i.e.* by starting from negative values. Second, it is exactly the fact that 0 belongs to the border (and the relative interior) of $\partial(f + g)$ that put the breaking point at 0 which is the fixed point of the represented operator. ◀

Remark 3.20 (Other proximal methods). The reasoning laid out above can be straightforwardly extended to other proximal methods of the literature. For instance, one can show identification for Douglas-Rachford/ADMM (Liang et al., 2017b), SAGA/SVRG (Poon et al., 2018), proximal (quasi-)Newton methods (Lee, 2020), etc. ◀

3.5 Harnessing nonsmoothness

To conclude this chapter, let us put the presented results into perspective. In the introduction to Part A, we motivated our study by considering regularized learning problems and the identification of their prior structure. In this chapter, we modeled this (user-defined) prior as a collection of manifolds:

$$\mathrm{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}.$$

Then, benefiting from some structure amounts to belonging to one or several of these manifolds. Importantly, we also saw that when computating a proximity operator, we often obtained the structure of its output (its sparsity pattern for the ℓ_1 norm, its rank for the nuclear norm, etc.).

In a first time, we showed how to guarantee that the iterates of a proximal method $x_k \in \mathrm{prox}_{\gamma g}(y_k)$ will belong to some of these manifold in finite time. More precisely, an iterate x_k will belong to a manifold $\mathcal{M}_i$ if it is a stable output of the proximity operator, *i.e.* if $\mathrm{prox}_{\gamma g}(y) \in \mathcal{M}_i$ for all y close to y_k. In particular, if y_k converges to some $\overline{y}$, the stable manifolds around $\overline{y}$ will be identified for sure while the other manifolds to which $\overline{x} = \mathrm{prox}_{\gamma g}(\overline{y})$ does not belong will not be identified. This also means that parts of the structure may be identified sooner than others. Some manifolds can even be falsely identified but only far enough from the optimal point. This proximal identification is the theoretical backbone to the design and the analysis of methods that are able to adapt to the uncovered structure along the way.

Then, we provided qualification condition for the exact identification of the final/optimal structure $\mathcal{M}^\star$. This additional qualification condition is necessary here since the optimal structure has to be somehow stable to be identified as we saw in Remark 3.3 and Fig. 3.1; this also makes sense in terms of statistical relevance of the optimal pattern.

In a second time, under the umbrella of partial smoothness, we investigated the identification of the optimal manifold $\mathcal{M}^\star$ under a qualification condition. This enabled

us to show that proximal methods not only identified but also behaved smoothly around the critical points. This means a possibly faster convergence in practice but also the opportunity to take advantage of this gained smoothness to use higher order methods along the identified structure.

These results are the workhorses and the motivation carrying Chapters 4–6.

4 Case Study: the Proximal Gradient method and its accelerated versions

À la fin, quand il vit
Que l'autre touchait presque au bout de la carrière,
Il partit comme un trait ; mais les élans qu'il fit
Furent vains : la Tortue arriva la première.
"Eh bien, lui cria-t-elle, avais-je pas raison ?
De quoi vous sert votre vitesse ?
Moi l'emporter ! et que serait-ce
Si vous portiez une maison ?
Jean de la Fontaine – *Le Lièvre et la Tortue (1668)*

In this chapter, we focus on some of the (many) variants of the proximal gradient method and investigate the identification properties presented in Chapter 3. Notably, we investigate how the difference in monotonicity and rates between the variants may result in contrasting identification in practice.

This chapter is based on the following publications:

- F. Iutzeler, J. Malick : On the Proximal Gradient Algorithm with Alternated Inertia , Journal of Optimization Theory and Applications, vol. 176, no. 3, pp. 688-710, March 2018.
- G. Bareilles, F. Iutzeler : On the Interplay between Acceleration and Identification for the Proximal Gradient algorithm , Computational Optimization and Applications, vol. 77, no. 2, pp. 351–378, 2020.

As mentioned in Part A's introduction, problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x) \qquad (\mathcal{P}_{\mathrm{A}})$$

with f a smooth function and g a nonsmooth, structure-enhancing, function are of particular interest in data science. In order to take full advantage of the structure of the problem, *i.e.* using the gradient of f and the proximity operator of g, it is natural to consider the proximal gradient algorithm. This method stems from the idea of Forward-Backward splitting in monotone operators theory (see (Bruck Jr, 1977; Passty, 1979) for an historical perspective and for a more recent formulation (Bauschke and Combettes, 2011, Chap. 25.3,27.3)). It consists in alternating a gradient step on f and a proximal operator on g. The proximal gradient has been widely adopted in signal/image processing and machine learning thanks to the availability of explicit proximity operator for regularization functions (Combettes and Wajs, 2005; Duchi and

Singer, 2009). The structure identification was also one of the reasons for the adoption of these kind of methods, in particular for the iterative thresholding of wavelets coefficients; see *e.g.* (Chambolle et al., 1998; Daubechies et al., 2004; Figueiredo and Nowak, 2003; Figueiredo et al., 2007) and references therein.

In 2009, Amir Beck and Marc Teboulle introduced an accelerated variant of the proximal gradient algorithm, FISTA (Beck and Teboulle, 2009a), that adapts the idea of Nesterov's fast gradient method (Nesterov, 1983) to the proximal gradient case; see (d'Aspremont et al., 2021) for a recent review. Thanks to its better convergence rate in theory and in practice and the relative simplicity of the algorithm, FISTA was rapidly adopted by the community and lead to a great amount of subsequent research. In particular, two aspects of FISTA caught my attention:

(1) The iterates convergence is more involved. The issue comes from the difficulty to mix the usual proofs based on monotone operator theory with Nesterov's acceleration; see (Iutzeler and Hendrickx, 2019, Sec. 3.2.1) and references therein. It was finally proven for slightly modified version in (Attouch and Peypouquet, 2016; Chambolle and Dossal, 2015).

(2) The functional decrease is not monotonous in general. In the follow-up paper (Beck and Teboulle, 2009b), Beck and Teboulle notice that "*As opposed to [the proximal gradient], FISTA is not a monotone algorithm, that is, the function values are not guaranteed to be nonincreasing. Monotonicity seems to be a desirable property of minimization algorithms, but it is not required in the original proof of convergence of FISTA. Moreover, we observed in the numerical simulations in (Beck and Teboulle, 2009a) that the algorithm is "almost monotone", that is, except for very few iterations the algorithm exhibits a monotonicity property.*" This was the motivation for their introduction of MFISTA, a monotone version of FISTA.

In fact, we noticed in (Bareilles and Iutzeler, 2020) that the non-monotonicity of the functional values was closely link to fluctuations of the identified structure. Intuitively, the convergence was too fast (more precisely, the inertia was too strong) to stay on some manifold, even if it is the optimal one; a problem that is not solved by the monotonous version MFISTA. This chapter is thus devoted to the study of the identification properties of proximal gradient algorithms.

4.1 The Proximal Gradient algorithm and its accelerated variants

In the convex case, a minimizer of problem ($\mathcal{P}_A$) is a point x satisfying

$$0 \in \partial F(x) = \nabla f(x) + \partial g(x). \tag{4.1}$$

Forward-Backward splitting then consist in numerically solving (4.1) by discretizing the differential inclusion $\dot{x} \in \partial F(x)$ in a Explicit-Implicit manner:

$$\begin{aligned} & 0 \in \gamma \nabla f(x_k) + \gamma \partial g(x_{k+1}) + x_{k+1} - x_k \\ \Leftrightarrow \quad & 0 \in \partial g(x_{k+1}) + \frac{1}{\gamma}\left(x_{k+1} - (x_k - \gamma \nabla f(x_k))\right) \\ \Leftrightarrow \quad & x_{k+1} = \operatorname{prox}_{\gamma g}\left(x_k - \gamma \nabla f(x_k)\right) \end{aligned}$$

and thus consists in alternating a proximal step and a gradient step.

In the whole chapter, we will thus assume convexity and non-degeneracy in order to make the message clearer.

Assumption 4.1. The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth, and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is convex, proper, and lower semi-continuous. Furthermore, we suppose that $\operatorname{argmin} F \neq \emptyset$.

4.1.1 Algorithm

For some fixed stepsize γ,[26] the proximal gradient algorithm consists in iterating

$$x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k)) \,. \qquad \text{(Proximal gradient)}$$

As mentioned in Section 2.3.3, this methods can be seen as an iterative minimization of a first-order approximation of f plus g since

$$\begin{aligned} x_{k+1} &= \operatorname{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k)) \\ &= \operatorname{argmin}_{u \in \mathbb{R}^n} \left\{ f(x_k) + \langle u - x_k, \nabla f(x_k) \rangle + g(u) + \frac{1}{2\gamma} \|u - x_k\|^2 \right\} \end{aligned}$$

which enables to easily derive descent inequalities (see Lemma 2.39) and prove the following convergence result.

Theorem 4.2. *Let Assumption 4.1 hold. Then, the Proximal gradient method with $\gamma \in (0, 1/L]$ generates iterates that verify*

a) $F(x_{k+1}) \leq F(x_k) - \frac{2-\gamma L}{2(1+\gamma L)} \operatorname{dist}(0, \partial F(x_{k+2}))^2$;

b) $F(x_k) - \min F \leq \frac{\|x^\star - x_0\|^2}{2\gamma k}$;

c) (x_k) *converges to a point* $\overline{x} \in \operatorname{argmin} F$;

d) $\|x_{k+1} - x\| \leq \|x_k - x\|$ *for any* $x \in \operatorname{argmin} F$.

In words, this means that the functional suboptimality decreases monotonically at a $O(1/k)$ rate. Furthermore, the iterates converge to one of the minimizers while the distance to the set of minimizers is non-increasing. This property, that often appears in monotone operator theory, is usually called *Fejér monotonicity* (Bauschke and Combettes, 2011, Sec. 5.1).

4.1.2 Acceleration by Nesterov's inertial method

To improve the convergence of the proximal gradient algorithm, (Beck and Teboulle, 2009a) proposed to add an *inertial* step. This consists in constructing the next iterate y_{k+1} by combining the outputs x_{k+1} and x_k of the last two proximal gradient steps. Specifically, given a sequence of real non-negative numbers (α_k), an iteration of the inertial proximal gradient algorithm can be written as

$$\begin{cases} x_{k+1} = & \operatorname{prox}_{\gamma g}(y_k - \gamma \nabla f(y_k)) \\ y_{k+1} = & x_{k+1} + \alpha_{k+1}(x_{k+1} - x_k) \end{cases} . \qquad \text{(Accelerated Proximal gradient)}$$

The design of the inertial sequence (α_k) affects greatly the performance of the resulting algorithm, and different options are investigated in the literature. Popular choices include a) Nesterov's optimal sequence (Nesterov, 1983) which leads to the FISTA algorithm (Beck and Teboulle, 2009a) and b) the variants used for refined analyses (Attouch and Peypouquet, 2016; Chambolle and Dossal, 2015):

$$\alpha_{k+1} = \frac{t_k - 1}{t_{k+1}} \quad \begin{array}{lllll} \text{with} & a) & t_0 = 0 & \text{and} & t_{k+1} := \frac{1+\sqrt{1+4t_k^2}}{2} \\ \text{or} & b) & t_0 = 0 & \text{and} & t_{k+1} := \frac{k+a}{a} \quad \text{with } a > 2. \end{array} \tag{4.2}$$

[26] For clarity and conciseness, we consider fixed stepsizes in the whole chapter. We refer the reader to (Beck, 2017, Chap. 10) for other stepsize strategies.

Both choices lead to increasing sequences tending to 1 at rate $1/k$ and are proven to accelerate the worst-case convergence rate of the algorithm from $O(1/k)$ to $O(1/k^2)$. However, this comes at the price of losing monotonicity.

Theorem 4.3. *Let Assumption 4.1 hold. Then, the Accelerated Proximal gradient method with $\gamma \in (0, 1/L]$ and (α_k) as in (4.2) generates iterates that verify*

a) ~~$F(x_{k+1}) \leq F(x_k)$~~*;*

b) $F(x_k) - \min F \leq \frac{\|x^\star - x_0\|^2}{2\gamma(k+1)^2}$*;*

c) (x_k) *converges to a point* $\overline{x} \in \operatorname{argmin} F$ *for choice b in (4.2);*

d) ~~$\|x_{k+1} - x\| \leq \|x_k - x\|$ for any $x \in \operatorname{argmin} F$~~.

Proof. Point b comes from (Beck and Teboulle, 2009a, Th. 4.4) using a very direct and elegant proof. Showing point c is more involved and the question remained open for more than five years,[27] see (Chambolle and Dossal, 2015, Th. 3) or (Attouch and Peypouquet, 2016, Th. 3) for two proofs based on different arguments. □

[27] *un lustre* in French!

The key idea in the proof of (Beck and Teboulle, 2009a, Th. 4.4) is that by calling the descent inequalities for the proximal gradient (in particular (2.15) with $x = u = y_k$ and $x = y_k, u = x^\star$), with $x^\star \in \operatorname{argmin} F$ and $v_k = F(x_{k+1}) - F(x^\star)$,

$$v_k - v_{k-1} = F(x_{k+1}) - F(x_k) \leq -\frac{2-\gamma L}{2\gamma}\|x_{k+1} - y_k\|^2 - \frac{1}{\gamma}\langle y_k - x_k, x_{k+1} - y_k\rangle$$
$$v_k = F(x_{k+1}) - \min F \leq -\frac{2-\gamma L}{2\gamma}\|x_{k+1} - y_k\|^2 - \frac{1}{\gamma}\langle y_k - x^\star, x_{k+1} - y_k\rangle$$

The first equation times $(t_k - 1)$ added to the second yields:

$$t_k v_k - (t_k - 1)v_{k-1} \leq -\frac{2-\gamma L}{2\gamma} t_k \|x_{k+1} - y_k\|^2 - \frac{1}{\gamma}\langle t_k y_k - (t_k - 1)x_k - x^\star, x_{k+1} - y_k\rangle.$$

Multiplying by t_k, using the relation $t_k^2 - t_k \leq t_{k-1}^2$ verified by both strategies in (4.2),

$$t_k^2 v_k - t_{k-1}^2 v_{k-1} \leq -\frac{2-\gamma L}{2\gamma}\|t_k x_{k+1} - t_k y_k\|^2 - \frac{1}{\gamma}\langle t_k y_k - (t_k - 1)x_k - x^\star, t_k x_{k+1} - t_k y_k\rangle$$

and thus

$$t_k^2 v_k - t_{k-1}^2 v_{k-1} \leq -\frac{1-\gamma L}{2\gamma}\|t_k x_{k+1} - t_k y_k\|^2 \tag{4.3}$$
$$- \frac{1}{2\gamma}\underbrace{\|t_k x_{k+1} - (t_k - 1)x_k - x^\star\|^2}_{:= b_{k+1}} + \frac{1}{2\gamma}\underbrace{\|t_k y_k - (t_k - 1)x_k - x^\star\|^2}_{:= a_k}.$$

The final step consists in noticing that this is a telescopic sum provided that $b_{k+1} = a_{k+1}$, ie.

$$t_k x_{k+1} - (t_k - 1)x_k = t_{k+1} y_{k+1} - (t_{k+1} - 1)x_{k+1} \Leftrightarrow y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k).$$

4.1.3 The Value of Monotonicity

The iterates generated by the Accelerated Proximal gradient are not monotonous in terms of functional values nor Fejér monotonous. In fact, the iterates generated by inertial methods can circle or oscillate around the set of minimizers (Lorenz and Pock, 2014; Maingé, 2008). These kinds of behaviors make accelerated methods sometimes slower than their unaccelerated counterparts (see for instance the non-negative least squares problem in (Malitsky and Pock, 2018)).

Monotonicity (and in particular functional monotonicity) is a desirable feature in optimization, in both theory and practice; see for instance the quests for descent in (Correa and Lemaréchal, 1993; Fuentes et al., 2012). For composite optimization, several algorithms based on descent tests have been proposed to fix the non-monotonicity of accelerated proximal gradient methods, in particular MTwist (Bioucas-Dias and Figueiredo, 2007) in the particular case of a quadratic f, and in the present setting MFISTA (Beck and Teboulle, 2009b):

$$\left\{\begin{array}{ll} z_{k+1} = & \operatorname{prox}_{\gamma g}\left(y_k - \gamma \nabla f(y_k)\right) \\ x_{k+1} = & \operatorname{argmin}\{F(x) : x \in \{z_{k+1}, x_k\}\} \\ y_{k+1} = & x_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_{k+1} - x_k) + \left(\frac{t_k}{t_{k+1}}\right)(z_{k+1} - x_{k+1}) \end{array}\right. \tag{MFISTA}$$

with (t_k) as in (4.2).

This direct modification enables to restore monotonicity of the functional values while preserving the convergence rate.

Theorem 4.4. *Let Assumption 4.1 hold. Then, the MFISTA method with $\gamma \in (0, 1/L]$ and (t_k) as in (4.2) generates iterates that verify*

a) $F(x_{k+1}) \leq F(x_k)$;

b) $F(x_k) - \min F \leq \frac{\|x^\star - x_0\|^2}{2\gamma(k+1)^2}$;

c) ~~*(x_k) converges to a point $\overline{x} \in \operatorname{argmin} F$ for choice b in (4.2);*~~

d) ~~$\|x_{k+1} - x\| \leq \|x_k - x\|$ *for any* $x \in \operatorname{argmin} F$.~~

Proof. Point a is immediate from the algorithm, and b comes from (Beck and Teboulle, 2009b, Th. 5.1) by a small modification of the proof of FISTA. I was not able to find a positive nor a negative answer in the literature for the iterates convergence of MFISTA, however tying to adapt the proof of (Chambolle and Dossal, 2015) to this version would give a good conjecture of the answer. □

Another strategy coming from monotone operator theory is to alternate between an accelerated proximal gradient step and a classical one. It was shown to exhibit attractive performances in practice (Iutzeler and Hendrickx, 2019) (better than inertial acceleration in some strongly convex cases), while recovering some iterate monotonocity. This method simply consists in applying Accelerated Proximal gradient with an alternated inertial sequence:

$$\alpha_k = 0 \tag{4.4}$$

$$\alpha_{k+1} = \frac{t_k - 1}{t_{k+1}} \quad \begin{array}{lllll} \text{with} & a) & t_0 = 0 & \text{and} & t_{k+1} := \frac{1+\sqrt{1+4t_k^2}}{2} \\ \text{or} & b) & t_0 = 0 & \text{and} & t_{k+1} := \frac{k+a}{a} \quad \text{with } a > 2. \end{array}$$

for k even. We obtain the following result in (Iutzeler and Malick, 2018).

Theorem 4.5. *Let Assumption 4.1 hold. Then, the Accelerated Proximal gradient method with $\gamma \in (0, 1/L]$ and (α_k) as in (4.4) generates iterates that verify*

a) $F(x_{k+2}) \leq F(x_k) - \frac{(2-\alpha_{k+1}-\gamma L)\gamma}{2(1+\gamma L)^2} \operatorname{dist}(0, \partial F(x_{k+2}))^2$ *for k even;*

b) ~~$F(x_k) - \min F \leq \frac{\|x^\star - x_0\|^2}{2\gamma(k+1)^2}$~~;

and if $0 \leq \alpha_k \leq \min\left\{\frac{1}{2}, \frac{1}{\gamma L} - \frac{1}{2}\right\}$ *for all k, then*

c) (x_{2k}) *converges to a point* $\overline{x} \in \operatorname{argmin} F$*;*

d) $\|x_{k+2} - x\| \leq \|x_k - x\|$ *for any* $x \in \operatorname{argmin} F$ *for k even.*

Proof. This is exactly Theorems 3.1 and 3.2 in (Iutzeler and Malick, 2018). For points c and d, convergence of the iterates may still hold for a wider choice of (α_k) but the convergence will not be Fejér monotonous anymore. □

Unlike MFISTA, no functional evaluation is needed to guarantee the functional monotonicity. In addition, even though we lose the $O(1/k^2)$ convergence rate, we will see in the next section that a descent as in point i also has a theoretical interest as it opens the door for a complete complexity analysis.

Remark 4.6 (Other variants & Kurdyka-Łojasiewicz functions). Many variants of the accelerated proximal gradient have been proposed since (Beck and Teboulle, 2009a); see *e.g.* (Li and Lin, 2015) for an overview. Many of them are designed to handle nonconvex functions. To perform a convergence rate analysis in this case, it is common to rely on the geometric properties of the function at hand, modeled by Kurdyka-Łojasiewicz gradient inequalities; see *e.g.* the extended survey in (NGuyen, 2017). Interestingly, these kind of properties also provide rates for subgradient descent methods such as our variant with alternated inertia; this was the viewpoint we adopted in (Iutzeler and Malick, 2018). ◀

4.1.4 Monotonic convergence and rates for Kurdyka-Łojasiewicz functions

By comparing the functional decrease of MFISTA (Theorem 4.4) with the one of the vanilla proximal gradient or the proximal gradient with alternated inertia (Theorems 4.2 and 4.5), we notice that MFISTA is "only" non-increasing whereas the latter methods decrease their functional value by an amount proportional to the $\operatorname{dist}(0, \partial F(x_{k+1}))^2$. This kind of monotonicity can be combined with geometric properties of objective functions to derive convergence rates or complexity analysis.

Two types of geometric profiles are often used: error bounds or Kurdyka-Łojasiewicz gradient inequalities. For a function $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ and all x close to $\operatorname{argmin} F$, these properties can be written as

$$\varphi(F(x) - \min F) \geq \operatorname{dist}(x, \operatorname{argmin} F) \qquad \text{(Error Bound)}$$
$$\varphi'(F(x) - \min F) \operatorname{dist}(0, \partial F(x)) \geq 1 \qquad \text{(Kurdyka-Łojasiewicz)}$$

where φ is a smooth increasing concave function, called *desingularizing function*; see (Bolte et al., 2007) or (Bolte et al., 2015) for details.

Typical desingularizing functions are of the form $\varphi(t) = C/\theta\, t^\theta$ for $\theta \in (0, 1]$ and $C > 0$. Interestingly, in this case, the two properties are equivalent for convex functions and share the same desingularizing function (Bolte et al., 2015, Th. 5). We

can also rephrase the property in a simpler way: F has the KL property if for all x close to $\operatorname{argmin} F$

$$\operatorname{dist}(0, \partial F(x)) \geq \frac{1}{C}(F(x) - \min F)^{1-\theta}. \tag{4.5}$$

This property may look strong at a first sight but it turns out to be widely satisfied, for instance by semi-algebraic functions (Bolte et al., 2007). In some special cases, the parameters θ and C can be explicitly computed; see (Bolte et al., 2015, Sec. 3). For instance, the ℓ_1-regularized least-square functions of the form

$$F(x) = \|Ax - b\|_2^2 + \lambda_1 \|x\|_1$$

has the KL property on any ℓ_1-ball around the solutions with $\varphi(s) = C\sqrt{s}$ and C computable explicitly from A, b, λ_1 and the size of the ball (Bolte et al., 2015, Lem. 10).

The Kurdyka-Łojasiewicz property is particularly well-suited for complexity analysis of algorithms showing a decrease of the form[28]

$$F(x_{k+1}) \leq F(x_k) - a_k \ \operatorname{dist}^2(0, \partial F(x_{k+1}))$$

see in particular (Attouch and Bolte, 2009; Frankel et al., 2015) or (Li and Lin, 2015) in the context of accelerated proximal gradient.

While the case of $a_k \geq a > 0$ is well covered in the literature, it does not suffices to analyze our algorithm with alternated inertia (see Theorem 4.5). We thus extended the result of (Attouch and Bolte, 2009) to incorporate the case of vanishing (a_k) in (Iutzeler and Malick, 2018, Th. 3.3).

[28] This is closely related to the notion of *subgradient sequences* considered in (Bolte et al., 2015): $F(x_{k+1}) \leq F(x_k) - a\|x_{k+1} - x_k\|^2$ and $\|v_{k+1}\| \leq b\|x_{k+1} - x_k\|$ for some $v_{k+1} \in \partial F(x_{k+1})$. Unfortunately, our alternated inertial methods do not fall into that framework. In particular, this prevents us to deduce anything on the iterates behavior.

Theorem 4.7. *Let $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a function satisfying the Kurdyka-Łojasiewicz inequality with perspective function $\varphi(t) = \frac{C}{\theta} t^\theta$ for some $C > 0$, $\theta \in (0, 1]$. Suppose that there is a non-negative sequence (a_k), such that the iterates (x_k) satisfy*

$$F(x_{k+1}) \leq F(x_k) - a_k[\operatorname{dist}(0, \partial F(x_{k+1}))]^2 \quad \text{and} \quad \sum_{k=1}^{\infty} a_k = +\infty$$

then $F(x_k)$ converge to $\min F$, and depending on the behavior of (a_k) divided in three regimes:

a) $a_k \geq a > 0$

b) $a_k \geq 0$ with $a_k = \Omega\left(\frac{1}{k^d}\right)$, $d \in (0, 1)$

c) $a_k \geq 0$ with $a_k = \Omega\left(\frac{1}{k}\right)$

we get the following functional convergence rates:

	$\theta \in]0, 0.5[$	$\theta \in [0.5, 1[$	$\theta = 1$
a)	$O\left(\frac{1}{k^{1+\frac{2\theta}{1-2\theta}}}\right)$	$O\left(\left[\frac{C^2}{C^2+a}\right]^k\right)$	*finite*
b)	$O\left(\frac{1}{k^{1+\frac{2\theta-d}{1-2\theta}}}\right)$	$O\left(\exp\left(-\frac{C'}{2C^2}k^{1-d}\right)\right)$	*finite*
c)	$O\left(\frac{1}{\log(k)^{\frac{1}{1-2\theta}}}\right)$	$O\left(\frac{1}{k^{\frac{C''}{2C^2}}}\right)$	*finite*

with $C' = \liminf_k \sum_{\ell \leq k} a_\ell / k^{1-d}$ and $C'' = \liminf_k \sum_{\ell \leq k} a_\ell / \log(k)$.

Proof. The monotonicity of $(F(x_k))$ implies that $F(x_k) \to \overline{F}$. Since $\sum_{k=1}^{\infty} a_k = +\infty$, we have moreover that a subsequence of $(\operatorname{dist}(0, \partial F(x_{k+1})))$ vanishes. By (4.5), this yields that the corresponding subsequence of $(F(x_k))$ converges to $\min F$, hence $\overline{F} = \min F$.

Define now $r_k = F(x_k) - \min F$ and come back to the KL property (4.5) which reads

$$r_{k+1} \leq r_k - \frac{a_k}{C^2} r_{k+1}^{2-2\theta} \quad \Leftrightarrow \quad \frac{a_k}{C^2} \leq (r_k - r_{k+1}) r_{k+1}^{2\theta-2}. \tag{4.6}$$

We separate three cases with respect to θ.

Case $\theta = 1$. Eq. (4.6) becomes $r_{k+1} \leq r_k - \frac{a_k}{C^2}$, so by summing this inequality we get

$$r_{k+1} \leq r_0 - \frac{1}{C^2} \sum_{\ell=0}^{k} a_\ell$$

and as (a_k) is a non-negative sequence verifying $\sum_{k=1}^{\infty} a_k = +\infty$, there is $K' < \infty$, such that $\forall k \geq K'$, $\frac{1}{C^2} \sum_{\ell=0}^{k} a_k > r_0$ leading to $r_{k+1} = F(x_{k+1}) - \min F < 0$ which contradicts $\min F$ being the minimum of F. Thus, we must have $F(x_{k+1}) = \min F$ for all $k \geq K'$, *i.e.* finite convergence.

Case $\theta \in [0.5, 1)$. Here $0 < 2 - 2\theta \leq 1$. Since $r_k \to 0$, we have $r_k^{2-2\theta} \geq r_k$ for all $k \geq K'$. Hence

$$r_{k+1} \leq r_k - \frac{a_k}{C^2} r_{k+1} \Leftrightarrow r_{k+1} \leq \frac{1}{1 + \frac{a_k}{C^2}} r_k$$

which leads to different convergence modes depending of (a_k). If it is bounded away from zero, linear convergence arises (case a). Else, if $a_k \to 0$, then there is $K'' < \infty$ so that for all $k \geq K''$, $\log(\frac{1}{1+\frac{a_k}{C^2}}) \leq -\frac{a_k}{2C^2}$ so

$$\begin{aligned}
\log(r_{k+1}) &\leq \sum_{\ell=K''}^{k} \log\left(\frac{1}{1 + \frac{a_\ell}{C^2}}\right) + \log(r_0) \\
&\leq -\frac{1}{2C^2} \sum_{\ell=K''}^{k} a_\ell + \log(r_0) \leq \begin{cases} -\frac{C'}{2C^2} k^{1-d} + \log(r_0) & \text{case b} \\ -\frac{C''}{2C^2} \log(k) + \log(r_0) & \text{case c} \end{cases}
\end{aligned}$$

leading, for C_b, C_c two positive constants, to

$$r_{k+1} \leq \begin{cases} C_b \exp(-\frac{C'}{2C^2} k^{1-d}) & \text{case b} \\ C_c \frac{1}{k^{C''/(2C^2)}} & \text{case c} \end{cases}$$

Case $\theta \in (0, 0.5)$. Here $-2 < 2\theta - 2 < -1$ so as $0 \leq r_{k+1} \leq r_k$, we have $0 \leq r_k^{2\theta-2} \leq r_{k+1}^{2\theta-2}$.

Define $\phi(t) = \frac{C}{1-2\theta} t^{2\theta-1}$ with the same C, θ as in φ. Let us turn our attention to $\phi(r_{k+1}) - \phi(r_k)$ that we want to lower-bound by a constant. We proceed in two subcases:
If $r_{k+1}^{2\theta-2} \leq 2 r_k^{2\theta-2}$. Then we have

$$\begin{aligned}
\phi(r_{k+1}) - \phi(r_k) &= -\int_{r_{k+1}}^{r_k} \phi'(t) \mathrm{d}t = C \int_{r_{k+1}}^{r_k} t^{2\theta-2} \mathrm{d}t \geq C(r_k - r_{k+1}) r_k^{2\theta-2} \\
&\geq \frac{C}{2} (r_k - r_{k+1}) r_{k+1}^{2\theta-2} = \frac{a_k}{2C} := d_{a_k}
\end{aligned}$$

If $r_{k+1}^{2\theta-2} \geq 2r_k^{2\theta-2}$. Then we have $r_{k+1}^{2\theta-1} \geq 2^{\frac{2\theta-1}{2\theta-2}} r_k^{2\theta-1}$

$$\phi(r_{k+1}) - \phi(r_k) = \frac{C}{1-2\theta}(r_{k+1}^{2\theta-1} - r_k^{2\theta-1}) \geq \frac{C}{1-2\theta}\left(2^{\frac{2\theta-1}{2\theta-2}} - 1\right) r_k^{2\theta-1}$$
$$\geq \frac{C}{1-2\theta}\left(2^{\frac{2\theta-1}{2\theta-2}} - 1\right) r_0^{2\theta-1} := d_b$$

Thus, we have $\phi(r_{k+1}) - \phi(r_k) \geq \min(d_{a_k}, d_b) > 0$, thus for all $k \geq K$,

$$\phi(r_k) = \phi(r_K) + \sum_{\ell=K}^{k-1} \phi(r_{i+1}) - \phi(r_i) \geq \phi(r_K) + \sum_{\substack{\ell=K \\ d_{a_\ell} < d_b}}^{k-1} d_{a_\ell} + (k - K - K_a(k))d_b$$

$$\geq \frac{1}{2C} \sum_{\substack{\ell=K \\ d_{a_\ell} < d_b}}^{k-1} a_\ell + (k - K - K_a(k))d_b$$

where $K_a(k) := \text{Card}\left\{\ell \in [K,k] : d_{a_\ell} \leq d_b\right\}$. We have to split into two cases:
case a: $K_a(k) \to \bar{K_a} < +\infty$ and $\phi(r_k) \geq (k - K - \bar{K_a})d_b$ for all $k > K + \bar{K_a}$; hence the result from the classical case holds.
cases b and c: $K_a(k) \to +\infty$ and $K' := \sup\{\ell : d_{a_\ell} \geq d_b\}$ is finite as $a_k \to 0$. Then, for all $k > K'$

$$\phi(r_{k+1}) \geq \frac{1}{2C} \sum_{\ell=K'}^{k} a_\ell \geq \begin{cases} C'k^{1-d} & \text{case b} \\ C\text{"} \log(k) & \text{case c} \end{cases}$$

and thus

$$r_{k+1} \leq \begin{cases} \left(\frac{C}{(1-2\theta)C'k^{1-d}}\right)^{\frac{1}{1-2\theta}} & \text{case b} \\ \left(\frac{C}{(1-2\theta)C\text{"}\log(k)}\right)^{\frac{1}{1-2\theta}} & \text{case c} \end{cases}$$

which leads to the claimed result. □

This result directly provides rates for the Proximal gradient algorithm under case a, as observed in *e.g.* (Bolte et al., 2015). Case c on the other hand is able to provide convergence rates for the algorithm with alternated inertia as stated in the following theorem.

Corollary 4.8 (Convergence Rates with alternated inertia)**.** *Let Assumption 4.1 hold and suppose that F satisfies the Kurdyka-Łojasiewicz inequality with perspective function $\varphi(t) = \frac{C}{\theta}t^\theta$ for some $C > 0$, $\theta \in (0,1]$. Then, the Accelerated Proximal gradient method with $\gamma = 1/L$ and (α_k) as in (4.4) generates iterates that satisfy*

$$F(x_{2k}) - \min F \leq \begin{cases} O\left(\frac{1}{\log(k)^{\frac{1}{1-2\theta}}}\right) & \text{if } \theta \in]0, 0.5[\\ O\left(\frac{1}{k^p}\right) & \text{if } \theta \in [0.5, 1[\\ 0 \text{ for } k \text{ large enough} & \text{if } \theta = 1 \end{cases}$$

with $p = \liminf_k \sum_{\ell \leq k} \frac{(2-\alpha_{\ell+1}-\gamma L)\gamma}{4(1+\gamma L)^2 C^2 \log(k)}$.

Unfortunately, these rates are strictly worse that those of the vanilla proximal gradient. But, they can be better than those of the accelerated version. Furthermore, this gives us an original manner to be resilient to strong convexity as described in the next remark.

Remark 4.9 (Resilience to strong convexity). When F is in addition μ-strongly convex, the proximal gradient algorithm is known to have a linear convergence (see (Karimi et al., 2016) for a proof based on error bounds) but accelerated versions such as FISTA only have polynomial convergence ($O(1/k^2)$ in general). Moreover these convergence rates clearly appear in practice; see *e.g.* (Malitsky and Pock, 2018).

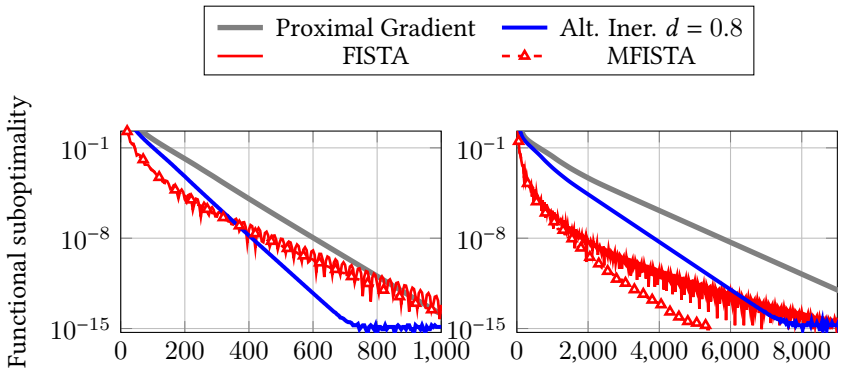
Since we can easily show that a strongly convex function verifies the KL property with $\theta = 0.5$, we can adopt an alternated inertial sequence of the form

$$\begin{aligned} \alpha_k &= 0 \\ \alpha_{k+1} &= \frac{t_k - 1}{t_{k+1}} \text{ with } \quad t_0 = 0 \text{ and } t_{k+1} := \left(\frac{k+a}{a}\right)^d \\ &\text{where } d \in (0,1) \text{ and } a > \max(1, (2d)^{1/d}). \end{aligned} \tag{4.7}$$

for k even. This "slower" inertial sequence was investigated (without alternance) in the completely different topic of inexact proximal computation in (Aujol and Dossal, 2015).

Then, assuming that Assumption 4.1 holds with F strongly convex. The Accelerated Proximal gradient method with $\gamma = 1/L$ and (α_k) as in (4.7) generates iterates that converge to the unique minimizer of F and have a better-than-polynomial rate of $O\left(\exp\left(-Ck^{1-d}\right)\right)$. ◀

4.1.5 Illustration



(a) 130×80 (b) 85×80

Figure 4.1: Illustration of the behavior of several variants of the proximal gradient algorithm on a lasso problem.

In order to visualize the typical phenomenons that we mentioned, we display the evolution of the variants of proximal gradient considered in this section on a lasso problem

$$\min_{x \in \mathbb{R}^n} \underbrace{\|Ax - b\|_2^2}_{f(x)} + \underbrace{\lambda \|x\|_1}_{g(x)}$$

with synthetic matrix/vector couples $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A is drawn from the standard normal distribution and $b = Ax_0 + e$ where x_0 is taken as a 10% sparse vector

taken from the normal distribution, and e is taken from the normal distribution with standard deviation 0.001. We set λ so that the original sparsity is ultimately recovered.

In Fig. 4.1, we plot the functional suboptimality for all compared algorithms with two different sizes of the matrix A: (a) 130×80 and (b) 85×80. We observe that the proximal gradient and the alternated inertial counterpart benefit from a more-than-polynomial rate which enables them to outperform other vairants in the better conditioned case, the alternated inertial version being always significantly better than the vanilla one. In the second case where the problem is more poorly conditioned, FISTA and MFISTA perform best.

4.2 Identification

Let us now turn to the identification properties of the (accelerated) proximal gradient in theory and in practice.

4.2.1 In theory: Identification and Acceleration are independent

Since the iterates produced by both the Proximal gradient and Accelerated Proximal gradient methods converge, the identification results of Chapter 3 give exactly the same result for these methods.

Theorem 4.10. *Let Assumption 4.1 hold. Assume that (x_k) is a sequence produced by either the Proximal gradient or the Accelerated Proximal gradient method with $\gamma \in (0, 1/L]$ and (α_k) as in (4.2). Provided that the limit $\overline{x}$ of (x_k) belongs to some manifold $\mathcal{M}$, if:*

1) there is $\varepsilon > 0$ such that

$$\text{for all } y \in \mathcal{B}(\overline{x} - \gamma \nabla f(\overline{x}), \varepsilon), \quad \operatorname{prox}_{\gamma g}(y) \in \mathcal{M}, \qquad \text{(PQC – Proximal Gradient)}$$

or 2) g is partly-smooth relative to $\mathcal{M}$ at $\overline{x}$ and

$$-\nabla f(\overline{x}) \in \operatorname{ri} \partial g(\overline{x}) \qquad \text{(SQC – Proximal Gradient)}$$

then, after some finite time $x_k \in \mathcal{M}$.

Proof. For Proximal gradient and Accelerated Proximal gradient, let us define the sequence (u_k) respectively as $u_k = x_k - \gamma \nabla f(x_k)$ and $u_k = y_k - \gamma \nabla f(y_k)$ so that $x_{k+1} = \operatorname{prox}_{\gamma g}(u_k)$ for both algorithms. Then, Theorems 4.2 and 4.3 give us the convergence of (u_k) to $\overline{u} = \overline{x} - \gamma \nabla f(\overline{x})$ with $\overline{x} \in \operatorname{argmin} F$ is the limit of (x_k).

The first part of the result is thus a direct application of Theorem 3.2 while the second one comes from Corollary 3.15. □

We notice here that the minimizer $\overline{x}$ has to be qualified for identification to happen, as already observed in the previous chapter. In addition, we see here that keeping the same minimizer, the shape of g around the minimizer plays an important role in the difficulty to identify some manifold.

Remark 4.11 (Faster rates around the solutions). Let us consider the second set of assumptions of Theorem 4.10, with g is partly-smooth relative to $\mathcal{M}$ at $\overline{x}$ and $-\nabla f(\overline{x}) \in \operatorname{ri} \partial g(\overline{x})$. Then, if f is additionally C^2 around $\overline{x}$ and $\ker(\nabla^2 f(\overline{x})) \cap T_{\overline{x}}\mathcal{M} = \{0\}$, a local linear convergence rate can be obtained. This is shown in (Liang et al., 2017a, Th. 4.13).

The same proof holds for both Proximal gradient and Accelerated Proximal gradient, it consists in showing that for both methods, denoting $r_k = x_k - \overline{x}$, $d_k = [r_{k-1} \quad r_k]^\top$:

$$d_{k+1} = Md_k + e_k, \qquad \|e_k\| = o(\|d_k\|)$$

with M a matrix of spectral radius $\rho(M) < 1$ from (Liang et al., 2017a, Cor. 4.9, Rem. 4.10).

In order to show the linear convergence behavior, following (Polyak, 1987, Chap. 2.1.2) (as in (Liang et al., 2017a)), we take $\varepsilon \in (0, (1-\rho(M))/2)$ and $K' > 0$ as the smallest time from which i) identification holds; and ii) $\|e_k\|/\|d_k\| \leq \varepsilon$. K' is finite since identification happens in finite time and $\|e_k\| = o(\|d_k\|)$. Then,

$$\|d_{K'+k+1}\| \leq \|M^k\|\|d_{K'+1}\| + \sum_{l=1}^{k} \|M^{k-l}\|\|e_{K'+l}\| \leq C_M\rho^k\|d_{K'+1}\| + C_M \sum_{l=1}^{k} \rho^{k-l}\|e_{K'+l}\|$$

where $\rho := \rho(M) + \varepsilon$ and $C_M > 0$ is a constant such that $\|M^k\| \leq C_M\rho^k$ for all k (see (Horn and Johnson, 2012, Cor. 5.6.13)).

This recursion enables us to show that there is some C such that for all $k \geq 0$, $\|d_{K'+k+1}\| \leq C(\rho(M) + 2\varepsilon)^k\|d_{K+1}\|$; see (Bareilles and Iutzeler, 2020, Apx. 5) for details.

Once again, the result is not different between Proximal gradient and Accelerated Proximal gradient. Even worse, the rate obtained for Accelerated Proximal gradient with this analysis is actually worse than the one for Proximal gradient (see (Liang et al., 2017a, Sec. 4.4)), which mirrors the discussion in Remark 4.9 above in the case where some strong convexity is present but unknown. ◂

4.2.2 *In practice: Interplay between Acceleration and Identification*

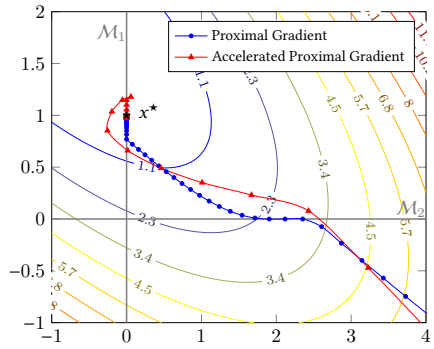
At this point, we saw that accelerated methods are favored in practice and offer better theoretical rates but these superior performances make no apparent difference in terms of identification. Nevertheless, we can expect accelerated methods to reach the basin of attraction of the optimal manifold sooner, and thus identify faster. However, the non-monotonicity of the accelerated methods may cause them to leave the identified manifold only to reach it again after some time. In this transient state, the interplay between identification and acceleration can be negative. Typically, an oscillatory behavior may be the sign of this transient state (see *e.g.* (Liang et al., 2017a, Sec. 5.4)).

Several works of the literature considered modifications of the Accelerated Proximal gradient aiming at limiting such a behavior using heuristic restarts (Catalina et al., 2018; Ito et al., 2017; O'donoghue and Candes, 2015; Scheinberg et al., 2014) or adaptive acceleration (Giselsson and Boyd, 2014; Poon and Liang, 2019); unfortunately, most of these results are empirical and lack a refined analysis. A notable exception is when the acceleration is limited to the iterations where the functional value decreases as mentioned in Section 4.1.3.

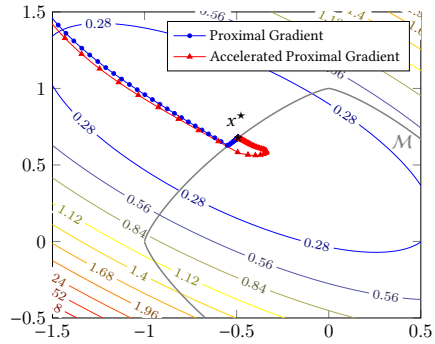
In (Bareilles and Iutzeler, 2020), we argue that acceleration can interfere with identification, sometimes delaying it, sometimes helping it. Note that we are only interested here in the transient phase of identification when the iterates are in the process of reaching the optimal manifold. We illustrate this claim by considering the iterates of Proximal gradient, Accelerated Proximal gradient, and MFISTA on three problems in $\mathbb{R}^2$ of the form $\|Ax - b\|^2 + g(x)$ for different nonsmooth functions g: the ℓ_1 norm and the distances to the unit ball in the 1.3 and 2.6 norms[29].

The natural manifolds of interest for these nonsmooth functions are respectively the set of cartesian axes of $\mathbb{R}^n$ and the unit spheres of the 1.3 and 2.6 norm, which

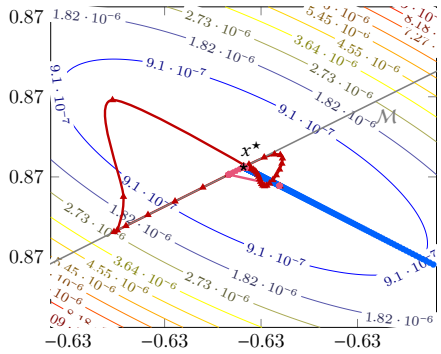
[29] $g = \min(\|\cdot\|_p - 1, 0)$ for p in $\{1.3, 2.6\}$, the proximity operator $\text{prox}_{\gamma g}(y)$ is $y(1 - \gamma/\|y\|_p)$ if $\|y\|_p > 1 + \gamma$, $y/\|y\|_p$ if $1 \leq \|y\|_p \leq 1 + \gamma$, and y otherwise.

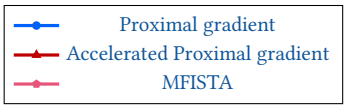


(a) $g(x) = \|x\|_1$ $\mathcal{M}_1 = y$-axis
$\mathcal{M}_2 = x$-axis (two linear manifolds)



(b) $g(x) = \max(0, \|x\|_{1.3} - 1)$
$\mathcal{M} = \mathcal{S}_{\|\cdot\|_{1.3}}(0, 1)$ (one curved manifold)





(c) $g(x) = \max(0, \|x\|_{2.6} - 1)$
$\mathcal{M} = \mathcal{S}_{\|\cdot\|_{2.6}}(0, 1)$ (one curved manifold)

Figure 4.2: Iterates behavior for variants of the proximal gradient algorithm when minimizing $\|Ax - b\|^2 + g(x)$ for different nonsmooth functions g. The candidate manifolds are displayed in gray.

we denote by $\mathcal{S}_{\|\cdot\|_{1.3}}(0, 1)$ and $\mathcal{S}_{\|\cdot\|_{2.6}}(0, 1)$. Figure 4.2 highlights three interesting behaviors:

(1) Upon reaching a manifold, the inertial term of the accelerated versions will not be aligned with the manifold in general and thus will have a non null orthogonal component to $\mathcal{M}$. Unless that orthogonal component is small enough, it will cause iterates to miss the manifold and go beyond it. Fig. 4.2a illustrates this point for accelerated proximal gradient over linear manifolds: the iterates go past the optimal manifold $\mathcal{M}_1$ twice before reaching it definitively, while the proximal gradient iterates identify it directly. Fig. 4.2c shows the same overshooting behavior of accelerated proximal gradient. In Fig. 4.2b, while proximal gradient iterates identify the optimal manifold definitively, iterates of both accelerated versions go beyond it, only to reach it again after several iterations.

(2) In the case of a curved (non-affine) manifold $\mathcal{M}$, the interplay between the curvature and the inertial term can cause the iterates to leave the manifold. In Fig. 4.2c, iterates of both accelerated versions reach $\mathcal{M}$ a first time but leave it after some iterations. It turns out that these iterates do reach $\mathcal{M}$ again, but only to leave it again some time later and have this phenomenon happen periodically.

(3) Acceleration also has some kind of exploratory behavior that increases the chances to encounter an optimal manifold, which can be helpful with problems not verifying the qualification conditions (see Theorem 4.10). In Fig. 4.2c where neither qualification condition hold, the iterates of both accelerated versions reach the optimal manifold, at least for some time, while proximal gradient iterates never does (this phenomenon is also illustrated in Remark 3.3).

As we already mentioned, identifying quickly the optimal structure of a composite optimization problem can be as important as solving the problem to a high precision, and even more. Thus, in the next section, we aim at conciliating acceleration with identification.

4.3 Improving the identification properties of the Accelerated Proximal gradient

Based on the previous remarks, it seems natural to try to accelerate as long as it does not jeopardize a fast identification. Our goal in this section is thus to provide an algorithm with a satisfying practical and theoretical rate (by accelerating as soon as possible) while making the iterates stick to the identified structure as much as possible.

4.3.1 A test-based strategy

We first start by laying out a generic accelerated proximal gradient where some test decides if an accelerated step should be performed or not. Next, we will propose two efficient practical tests to determine whether or not to accelerate an iteration with the goal of promoting sparsity.

At each iteration k, we call a boolean-valued function T_k that returns 1 if an acceleration step should be performed, and 0 otherwise. Our test-based algorithm then writes:

$$\begin{cases} y_k = \begin{cases} x_k + \alpha_k(x_k - x_{k-1}) & \text{if } \mathsf{T}_k = 1 \\ x_k & \text{otherwise} \end{cases} \\ u_{k+1} = y_k - \gamma \nabla f(y_k) \\ x_{k+1} = \operatorname{prox}_{\gamma g}(u_{k+1}) \end{cases} \qquad \text{(Test-dependent acceleration)}$$

A general bound on suboptimality can be derived for this algorithm, independently of the value of the test T, as stated in the following lemma.

Lemma 4.12. *Let Assumption 4.1 hold and take $\gamma > 0$. Then, the iterates produced by the proximal gradient method with (Test-dependent acceleration) and (α_k) as in (4.2) satisfy*

$$t_k^2 \left[F(x_{k+1}) - \min F\right] \leq -\sum_{\ell=0}^{k} \frac{1-\gamma L}{2\gamma} t_\ell^2 \|x_{\ell+1} - y_\ell\|^2 + \frac{1}{2\gamma}\|x_0 - x^\star\|^2 + \sum_{\ell=1}^{k} (1 - \mathsf{T}_\ell) \frac{1}{2\gamma}\|x_\ell - x^\star\|^2 \tag{4.8}$$

Proof. Following the same inequalities as for Accelerated Proximal gradient and considering the two outputs of the test:

- The accelerated update $\mathsf{T}_k = 1$ specifies (4.3) to:

$$t_k^2 v_k - t_{k-1}^2 v_{k-1} \leq -\frac{1-\gamma L}{2\gamma}\|t_k x_{k+1} - t_k y_k\|^2 \\ - \frac{1}{2\gamma}\|t_k x_{k+1} - (t_k - 1)x_k - x^\star\|^2 + \frac{1}{2\gamma}\|t_{k-1}x_k - (t_{k-1}-1)x_{k-1} - x^\star\|^2 \tag{4.9}$$

  since the update gives $t_k(y_k - x_k) = (t_{k-1} - 1)(x_k - x_{k-1})$.

- The proximal gradient update $\mathsf{T}_k = 0$ specifies (4.3) to:

$$t_k^2 v_k - t_{k-1}^2 v_{k-1} \leq -\frac{1-\gamma L}{2\gamma}\|t_k x_{k+1} - t_k y_k\|^2 \\ - \frac{1}{2\gamma}\|t_k x_{k+1} - (t_k - 1)x_k - x^\star\|^2 + \frac{1}{2\gamma}\|x_k - x^\star\|^2 \tag{4.10}$$

  since the update is $y_k = x_k$.

Both Eqs. (4.9) and (4.10) can be summarized, at the cost of introducing some error when acceleration is performed, as:

$$t_k^2 v_k - t_{k-1}^2 v_{k-1} \leq -\frac{1-\gamma L}{2\gamma}\|t_k x_{k+1} - t_k y_k\|^2 \\ - \frac{1}{2\gamma}\|t_k x_{k+1} - (t_k - 1)x_k - x^\star\|^2 + \frac{1}{2\gamma}\|t_{k-1}x_k - (t_{k-1}-1)x_{k-1} - x^\star\|^2 \\ + (1 - \mathsf{T}_k)\frac{1}{2\gamma}\|x_k - x^\star\|^2$$

A functional error bound can now be deduced, by summing these inequalities up from 1 to iteration k and re-arranging terms:

$$t_k^2[F(x_{k+1}) - \min F] \leq F(x_1) - \min F - \sum_{\ell=1}^{k} \frac{1-\gamma L}{2\gamma}\|t_\ell x_{\ell+1} - t_\ell y_\ell\|^2 \\ - \frac{1}{2\gamma}\|t_\ell x_{\ell+1} - (t_\ell - 1)x_\ell - x^\star\|^2 + \frac{1}{2\gamma}\|x_1 - x^\star\|^2 \\ + \sum_{\ell=1}^{k}(1 - \mathsf{T}_\ell)\frac{1}{2\gamma}\|x_\ell - x^\star\|^2 \tag{4.11}$$

where $t_0 = 1$. Suboptimality at first iteration can be approximated by applying a usual descent lemma (*i.e.* (2.15) with $x = y_0, u = x^\star$):

$$F(x_1) - \min F \leq -\frac{2-\gamma L}{2\gamma}\|x_1 - y_0\|^2 + \frac{1}{\gamma}\langle y_0 - x^\star, x_1 - y_0\rangle \\ = -\frac{1-\gamma L}{2\gamma}\|x_1 - y_0\|^2 + \frac{1}{2\gamma}\|y_0 - x^\star\|^2 - \frac{1}{2\gamma}\|x_1 - x^\star\|^2$$

Finally, recalling that $y_0 = x_0$, applying the previous majoration and $-\frac{1}{2\gamma}\|t_k x_{k+1} - (t_k - 1)x_k - x^\star\|^2 < 0$ to Eq. (4.11) yields the result. □

This results indicates us that the test cannot be equal to 0 (no acceleration) infinitely many times, otherwise we would lose our convergence rate. But, as we want identification to be favored, we also need a control on the iterates convergence.

This can be obtained using once again the geometric properties of the functions: since Error Bounds and KL inequalities are equivalent with the same desingularizing function (Bolte et al., 2015, Th. 5) (see also Section 4.1.4), if F satisfies (4.5), we obtain for any x close to $\operatorname{argmin} F$

$$\operatorname{dist}(0, \partial F(x)) \geq \frac{1}{C}(F(x) - \min F)^{1-\theta} \geq \frac{1}{C}\left(\frac{\theta}{C}\right)^{\frac{1-\theta}{\theta}} (\operatorname{dist}(x, \operatorname{argmin} F))^{\frac{1-\theta}{\theta}}.$$

In this case, using that $\operatorname{dist}(0, \partial F(x_{k+1})) \leq \frac{L\gamma+1}{\gamma}\|y_k - x_{k+1}\|$ (see Lemma 2.41), we get that

$$\|y_k - x_{k+1}\| \geq \frac{\gamma}{C(L\gamma+1)}\left(\frac{\theta}{C}\right)^{\frac{1-\theta}{\theta}} (\operatorname{dist}(y_k, \operatorname{argmin} F))^{\frac{1-\theta}{\theta}}, \tag{4.12}$$

which gives us the kind of control we need.

Furthermore, as we are mainly interested in final identification, it seems natural to do accelerated steps as long as the iterates are far away from the solution. Therefore, we only allow to consider non-accelerated steps when iterates live in the following set:

$$\mathsf{Z} = \left\{y : x = \operatorname{prox}_{\gamma g}(y - \gamma\nabla f(y)) \text{ satisfies } \|x - y\|^2 \leq \|x_1 - x_0\|^2 \text{ and } F(x) \leq F(x_0)\right\}.$$

To determine whether $y_k \in \mathsf{Z}$ or not, one just has to check if $\|x_{k+1} - y_k\|^2 \leq \|x_1 - x_0\|^2$ and $F(x_{k+1}) \leq F(x_0)$.[30] Thus, it depends only on previously computed iterates and not on the outcome of the test at time k.

[30] The constant $\|x_1 - x_0\|^2$ could actually be replaced by any constant; this one seems to work well in practice. Furthermore, the functional evaluation is actually only needed if the function is sharp with $\theta = 1$ above, *e.g.* when $f \equiv 0$ and $g(x) = \|x\|_1$. In these rather degenerate cases, the proximal gradient converges in a finite number of steps.

4.3.2 Test 1: Stopping when reaching

Let us define a set of candidate manifolds $\mathsf{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_q\}$ onto which identification is of particular importance. We refer the reader to Section 3.2.3 for discussions on the typical candidate manifolds in data science and how proximity operators can directly indicate if their output belong to some of these manifolds.

As noticed in Section 4.2.2 point (1), upon identification the momentum term is in general not aligned with the identified subspace. One further accelerated step may cause the next iterate to leave the subspace, while the vanilla proximal gradient would have stayed in it. A first natural method is thus to "reset" the inertial term, by performing one non-accelerated step when reaching a new manifold in our candidate set $\mathcal{C}$:

$$\mathsf{T}_k^1 = 0 \quad \text{(no acceleration)} \quad \text{if } y_k \in \mathsf{Z} \text{ and } \begin{cases} x_k \in \mathcal{M} \\ x_{k-1} \notin \mathcal{M} \end{cases} \quad \text{for some } \mathcal{M} \in \mathcal{C}$$

and 1 otherwise.

Note that this kind of test is computationally possible for most regularization functions in our applications of interest since the computation of the proximity operator comes with the knowledge of the structure of the output, see Sections 3.2.3 and 3.5.

Intuitively, acceleration is performed by default if the iterates are too far from optimum and as long as no new structure is identified. This means that we can benefit from the exploratory behavior of acceleration. Then, accelerated iterations will be performed as long as they do not prevent identifying a new manifold. Note that this way, if finite-time identification is possible, iterations should asymptotically all be accelerated. As expected, this method has the same rate of convergence as the

accelerated proximal gradient whenever identification is possible, which is proven in the following theorem. The interest of this method lies in its non-asymptotic *identification* behavior, not captured by theory, which should be improved compared with proximal and accelerated proximal gradient.

Theorem 4.13. *Let Assumption 4.1 hold, suppose that F satisfies the Kurdyka-Łojasiewicz inequality with perspective function $\varphi(t) = \frac{C}{\theta} t^\theta$ for some $C > 0$, $\theta \in (0, 1]$, and take $\gamma \in (0, 1/L]$. Then, the iterates produced by the proximal gradient method with* (Test-dependent acceleration) *endowed with T^1 and (α_k) as in 4.2 satisfy*

$$F(x_{k+1}) - \min F \leq \frac{\|x_0 - x^\star\|^2}{2\gamma t_k^2} + \frac{k\mathsf{R}}{2\gamma t_k^2} = O\left(\frac{1}{k}\right)$$

for any $x^\star \in \operatorname{argmin} F$ and some $\mathsf{R} > 0$.
Furthermore, if Problem ($\mathcal{P}_\mathrm{A}$) has a unique minimizer $x^\star$ and the qualifying constraint (PQC – Proximal Gradient) *holds at $x^\star$ for all $\mathcal{M} \in \mathcal{C}$ such that $x^\star \in \mathcal{M}$ at $u^\star = x^\star - \gamma \nabla f(x^\star)$, then the iterates sequence (x_k) converges, finite-time identification happens, and*

$$F(x_{k+1}) - \min F \leq \frac{\|x_0 - x^\star\|^2}{2\gamma t_k^2} + \frac{K\mathsf{R}}{2\gamma t_k^2} = O\left(\frac{1}{k^2}\right).$$

for some finite $K > 0$.

Proof. First, at any iteration k for which the test returned 0 ($\mathsf{T}_k^1 = 0$), we have $\|x_{k+1} - y_k\| \leq \|x_1 - x_0\|^2$ and $F(x_{k+1}) \leq F(x_0)$ as $y_k \in \mathsf{Z}$. Thus, using the fact that F is a KL function and the reasoning of (4.12), we have that $\|y_k - x^\star\|^2 = \|x_k - x^\star\|^2$ is bounded by some constant R (note that $x_{k+1} = y_{k+1}$ since $\mathsf{T}_k^1 = 0$). Dropping the first term of the right hand side of (4.8) and using that $t_k \geq c.k$ for any choice in (4.2), we get the first result.

Then, if there is a unique minimizer, the error bound for F (see (4.12)) along with the first part of the result tells us that $x_k \to x^\star$ and thus $y_k \to x^\star$. Since the qualifying constraint (PQC – Proximal Gradient) holds for all final manifolds (*i.e.* all $\mathcal{M} \in \mathcal{C}$ such that $x^\star \in \mathcal{M}$), it follows from Corollary 3.9 that x_k belongs to exactly the same manifolds as $x^\star$ after some finite time. All in all, this means that the test T^1 will produce non accelerated iterates only a finite number of times K which gives the second part of the result. □

This result means that we can devise a test-based acceleration of the proximal gradient that will benefit from the same functional convergence rate as the Accelerated Proximal gradient while favoring structured iterates.

4.3.3 Test 2: Prospective reach

Another method to deal with the negative effects of inertia when the additional term is misaligned with the local manifold is to compute one proximal gradient step forward to investigate which structure can be expected from the next iterate. Intuitively, if the iterate obtained after acceleration is at least as structured as the non-accelerated one, it is kept, otherwise the point obtained after the simple proximal gradient step is taken. This is done in order to counteract both issues (1) and (2) mentioned in Section 4.2.2

while still benefiting from the exploratory behavior of acceleration (see point (3) in Section 4.2.2).

$$\mathsf{T}^2_k = 0 \quad \text{(no acceleration)} \quad \text{if } y_k \in \mathsf{Z} \text{ and } \begin{cases} \mathsf{T}(x_k) \in \mathcal{M} \\ \mathsf{T}(x_k + \alpha_k(x_k - x_{k-1})) \notin \mathcal{M} \end{cases} \text{for some } \mathcal{M} \in \mathcal{C}$$

and 1 otherwise with $\mathsf{T}(x) = \operatorname{prox}_{\gamma g}(x - \gamma \nabla f(x))$.

This approach is further motivated by the desirable retraction property of the proximal (gradient) operator (see Section 3.4.3). However, a drawback of this test is the necessity to compute two proximal gradient steps (for the accelerated and the non-accelerated point), a bit like MFISTA. Similarly to the test T^1, we are able to show that T^2 provides at least a convergence similar to that of the Proximal gradient, and in cases when identification is possible, equivalent to that of the Accelerated Proximal gradient.

Theorem 4.14. *Let Assumption 4.1 hold, suppose that F satisfies the Kurdyka-Łojasiewicz inequality with perspective function $\varphi(t) = \frac{C}{\theta} t^\theta$ for some $C > 0$, $\theta \in (0, 1]$, and take $\gamma \in (0, 1/L]$. Then, the iterates produced by the proximal gradient method with Test-dependent acceleration endowed with T^2 and (α_k) as in (4.2) satisfy*

$$F(x_{k+1}) - \min F \leq \frac{\|x_0 - x^\star\|^2}{2\gamma t_k^2} + \frac{k\mathsf{R}}{2\gamma t_k^2} = O\left(\frac{1}{k}\right)$$

for any $x^\star \in \operatorname{argmin} F$ and some $\mathsf{R} > 0$.
Furthermore, if Problem ($\mathcal{P}_\mathrm{A}$) has a unique minimizer $x^\star$ and the qualifying constraint (PQC – Proximal Gradient) holds at $x^\star$ for all $\mathcal{M} \in \mathcal{C}$ such that $x^\star \in \mathcal{M}$ at $u^\star = x^\star - \gamma \nabla f(x^\star)$, then the iterates sequence (x_k) converges, finite-time identification happens, and

$$F(x_{k+1}) - \min F \leq \frac{\|x_0 - x^\star\|^2}{2\gamma t_k^2} + \frac{K\mathsf{R}}{2\gamma t_k^2} = O\left(\frac{1}{k^2}\right).$$

for some finite $K > 0$.

Proof. The proof is the same as the one of Theorem 4.13. Indeed, like T^1, T^2 is such that i) the test can return 0 only for bounded iterates; and ii) as soon as identification happens, the test returns 1 (*i.e.* acceleration). □

Before looking at the presented tests numerically, we mention that the same reasoning as in Remark 4.9 can be applied to obtain local linear convergence for the methods presented here.

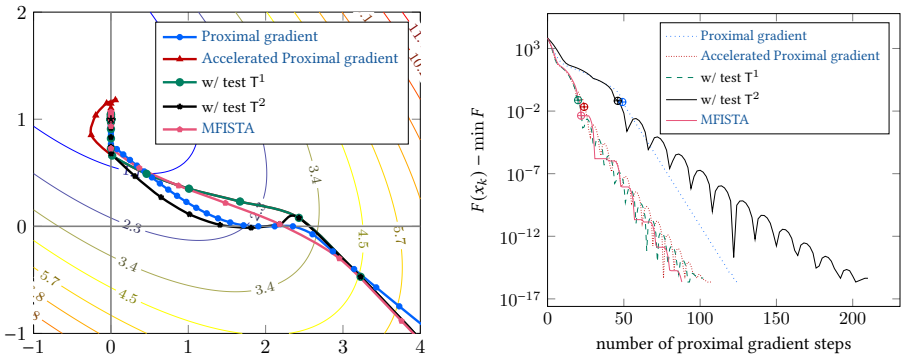
4.3.4 Numerical illustrations

In this section, we first show how the proposed methods can overcome the issues presented on test cases in Section 4.2.2. Then, we illustrate the improved identification properties of these methods on a more typical data science objective.[31]

[31] The code used for these experiments was written in Julia (Bezanson et al., 2017) by Gilles Bareilles and is available at https://github.com/GillesBareilles/Acceleration-Identification.

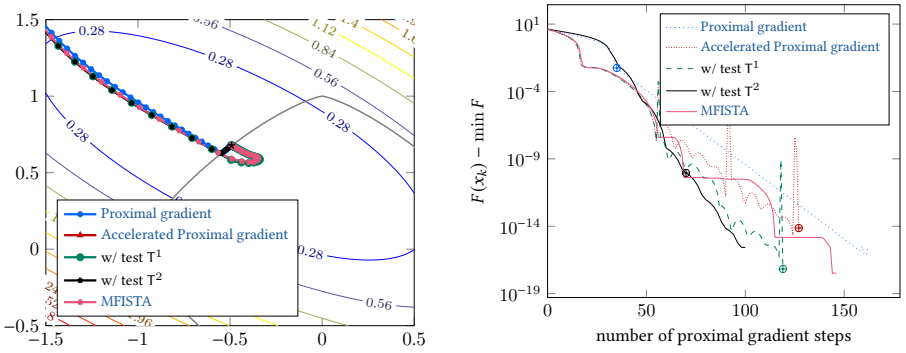
Test cases

First, we return on the test cases presented in Section 4.2.2, and show the iterates trajectories and suboptimality evolution along with the time of identification. For a



(a) Iterates behavior (b) Suboptimality

Figure 4.3: $F(x) = \|Ax - b\|^2 + g(x)$ with $g(x) = \|x\|_1$; $\mathcal{M}_1 = y$-axis and $\mathcal{M}_2 = x$-axis



(a) Iterates position (b) Suboptimality

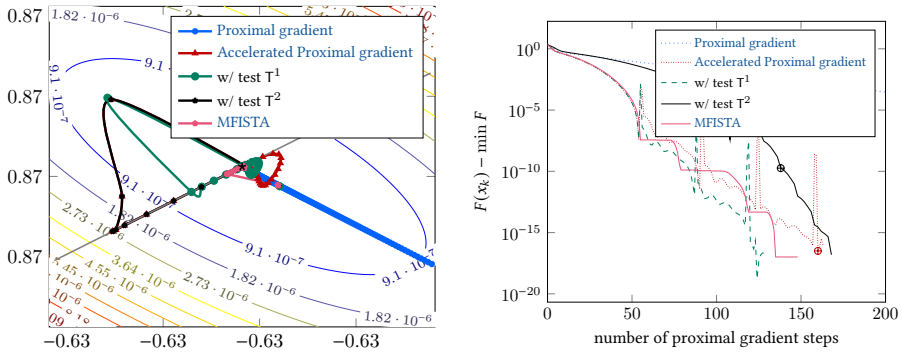
Figure 4.4: $F(x) = \|Ax - b\|^2 + g(x)$ with $g(x) = \max(0, \|x\|_{1.3} - 1)$; $\mathcal{M} = \mathcal{S}_{\|\cdot\|_{1.3}}(0, 1)$

fair comparison between test T^2 and the other algorithms, we plot the suboptimality versus the number of proximal gradient steps (equal to the number of iterations for all algorithms except with test T^2 which performs two proximal gradient steps per iteration). The moment of identification of the final structure is denoted by the symbol $\oplus$ on the suboptimality plots.

In Fig. 4.3, when g is the ℓ_1 norm, both tests allow to identify in finite time and prevent issue (1) of Section 4.2.2.

In Fig. 4.4 and Fig. 4.5, the sought manifolds are respectively the 1.3-norm and 2.6-norm unit sphere, which are curved. Test T^2 allows to get finite identification, while T^1 and accelerated proximal gradient struggle in doing so. Furthermore, the algorithm based on test T^2 identifies the manifold as soon as one of its iterates belong to it, as opposed to the accelerated proximal gradient or T^1.

All in all, we observe that test T^2 corrects the problems noted in Section 4.2.2 on these three examples. We advocate the use of T^2 when identifying the structure is most important. If reaching a high precision solution is the primary objective, we recommend to use test T^1, for which each iteration is as costly as an accelerated proximal gradient one.



(a) Iterates position (b) Suboptimality

Figure 4.5: $F(x) = \|Ax - b\|^2 + g(x)$ with $g(x) = \max(0, \|x\|_{2.6} - 1)$; $\mathcal{M} = \mathcal{S}_{\|\cdot\|_{2.6}}(0, 1)$

Low-rank matrix regression

Now, we consider a low-rank linear regression problem

$$\min_{X \in \mathbb{R}^{20\times 20}} \|AX - B\|_2^2 + \lambda \|X\|_*$$

where $A \in \mathbb{R}^{(16\times 16)\times(20\times 20)}$ is a random tensor whose coefficients follow a centered reduced normal distribution, $B = AS + E$ where S is a rank-3 matrix and E is a matrix with Gaussian entries with standard deviation 0.01. Finally, we set $\lambda = 0.01$ in order to retrieve the same structure as S (in the spirit of (Vaiter et al., 2017, Th. 1)).

We observe in Fig. 4.6 that the moment of identification happens roughly at the same time for the accelerated and proposed algorithms while the vanilla proximal gradient takes much more time. This justifies the use of acceleration in the first steps in order to explore correctly the search space. We also see that the number of correctly identified manifolds increases almost monotonically for T^2, while accelerated proximal gradient and T^1 seem to lose all structure upon identifying a new manifold. This means that if one stops all algorithms at a 10^{-3} suboptimality, almost no structure is recovered for the accelerated proximal gradient, while test T^1 and even more T^2 are able to recover half the structure of the original signal.

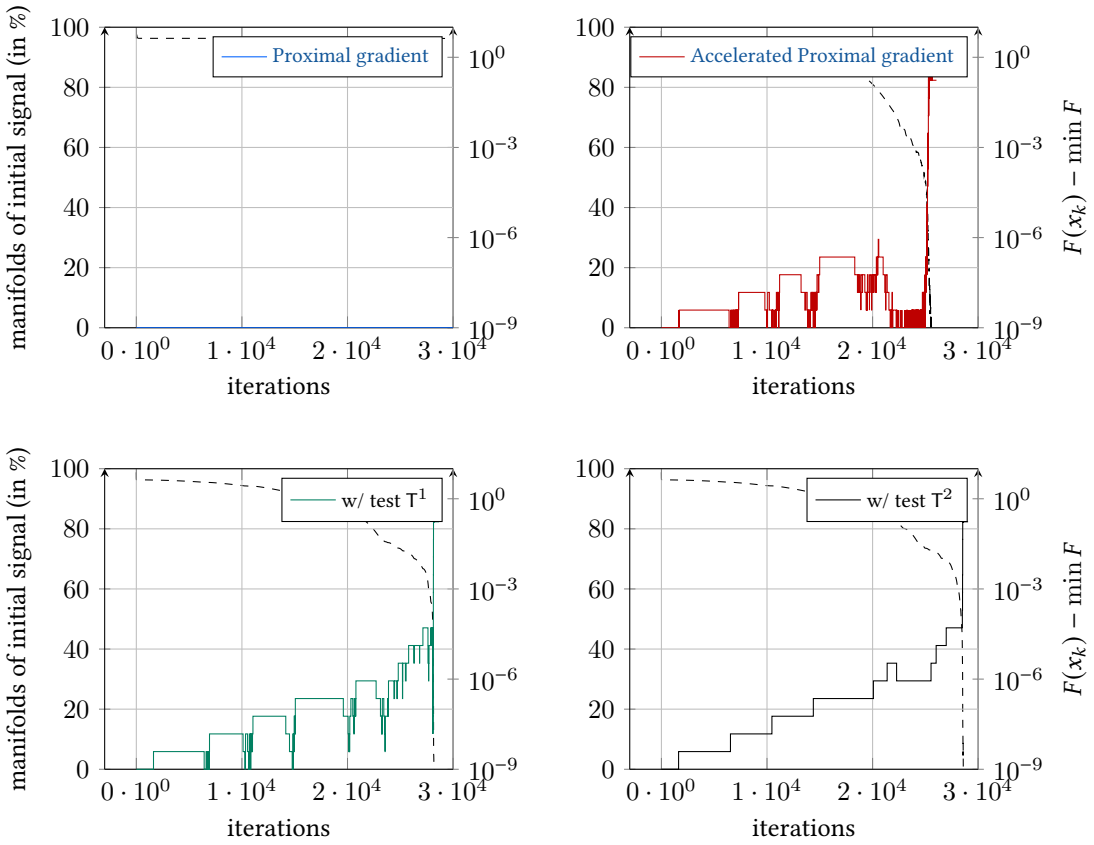


Figure 4.6: Low-rank linear regression – 20-3=17 manifolds to identify

4.4 Concluding remarks

In this chapter, we considered variants of the Proximal gradient algorithm, including the popular accelerated version FISTA. Since the identification properties rely only on the convergence of the iterates but not on any form of monotonicity (functional or Fejér), these methods appear to be equivalent with respect to structure identification in theory.

However, in practice, we showed that acceleration can interfere with identification, sometimes positively, sometimes negatively. Then, we proposed two simple modifications of the accelerated proximal gradient in order to counteract the negative effects of acceleration on the iterates structure during the transient phase when the structure is currently being identified. This was also the opportunity to show that the identification performance among the algorithms depends largely on whether the manifold is flat or has curvature.

More generally, this case study was the occasion to precisely show that while all converging proximal methods were equal for the identification results of Chapter 3, this is not the case in practice. Indeed, final identification holds for all methods but the *reaching time* and the *structure stability* can be very different. The reaching time is closely linked to the convergence rate of the iterates; as a rule of thumb the faster the rate, the faster some structure starts to appear. On the other hand, the structure stability very much depends on the algorithm and in particular on its relation with structure information. Typically, accelerated methods are often faster but tend to extrapolate iterates linearly, regardless of the identified structure. They are thus efficient on affine manifolds (like sparsity) but may be very unstable on general manifolds (such as fixed rank).

In the upcoming Chapters 5 and 6, we will show how the structure information can be exploited to a numerical advantage by directly exploiting its lower dimensionality.

5 Adaptive Coordinate Descent

HENRI– Donc voilà, Lucie hésite entre danseuse étoile,
star à Hollywood ou prof de philo.
On essaie de l'orienter subtilement vers la troisième option.
Alexis Michalik – *Le porteur d'histoire (1992)*

Building on the theory and intuitions developed in Chapters 3 and 4, we develop here a method that harnesses the uncovered structure of nonsmooth problems by exploring preferentially along this low-complexity manifold. To do so, in the case of linear priors, we introduce a randomized proximal gradient algorithm whose sampling depends on the identified structure. This brings a significant improvement on typical learning problems in terms of dimensions explored compared to comparable structure-blind methods.

This chapter is based on the following publication:

- D. Grishchenko, F. Iutzeler, J. Malick: Proximal Gradient methods with Adaptive Subspace Sampling, Mathematics of Operations Research, 2021.

As in Chapter 4, we will consider problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x) \tag{$\mathcal{P}_{\mathrm{A}}$}$$

with f a smooth function and g a nonsmooth structure-enhancing function, motivated by regularized signal processing problems as advocated in Part A's introduction. We will also work on variants of the Proximal gradient algorithm but with another viewpoint: reducing the dimension of the gradient update based on the structure identified by the proximity operator.

To do so, we first introduce a randomized descent algorithm going beyond separable nonsmoothness and associated coordinate descent methods: we consider "subspace descent" extending "coordinate descent" to generic subspaces. Then, we use the identification property of proximal methods to adapt our sampling of the subspaces with the identified structure. This results in a structure-adapted randomized method with automatic dimension reduction, which performs better in terms of dimensions explored compared to non-adaptive methods.

Though our main concern is with non-separable nonsmooth functions g, we mention that our identification-based adaptive approach is different from existing adaptation strategies restricted to the particular case of coordinate descent methods. Indeed, adapting coordinate selection probabilities is an important topic for coordinate descent methods as both theoretical and practical rates heavily depend on them (see

e.g. (Necoara and Patrascu, 2014; Richtárik and Takáč, 2014)). Though the optimal theoretical probabilities, named importance sampling, often depend on unknown quantities, these *fixed* probabilities can sometimes be computed and used in practice, see (Richtárik and Takáč, 2016b; Zhao and Zhang, 2015). The use of *adaptive* probabilities is more limited; some heuristics without convergence guarantees can be found in (Glasmachers and Dogan, 2013; Loshchilov et al., 2011), and greedy coordinates selection are usually expensive to compute (Dhillon et al., 2011; Nutini et al., 2015, 2017). Bridging the gap between greedy and fixed importance sampling, (Namkoong et al., 2017; Perekrestenko et al., 2017; Stich et al., 2017) propose adaptive coordinate descent methods based on the coordinate-wise Lipschitz constants and current values of the gradient. The methods we develop here, even when specialized in the coordinate descent case, are the first ones where the *iterates structure enforced by a non-smooth regularizer* is used to adapt the selection probabilities.

All along this chapter, we will consider linear structures and interpret our findings for sparsity. Nevertheless, the develop theory fits any kind of linear priors, some examples of which will be investigated into more details at the end of the chapter.

5.1 Randomized Subspace descent

The premise of randomized subspace descent consists in repeating two steps: i) randomly selecting some linear subspace; and ii) updating the iterate over the chosen subspace. Such algorithms thus extend usual coordinate descent to general sampling strategies, which requires algorithmic changes and an associated mathematical analysis. Note however that we restrict ourselves to linear subspaces.

We put the following standard assumption on the problem; compared to Chapter 4 on the proximal gradient, we also add a strong convexity assumption in order to simplify the claims and proofs. Proof directions for the non-strongly convex case are available in the appendix of (Grishchenko et al., 2020).

Assumption 5.1 (On the optimization problem)**.** The function f is L-smooth and μ-strongly convex and the function g is convex, proper, and lower-semicontinuous. This implies that F has a unique minimizer that we denote $x^\star$.

In the first part of this chapter, we will investigate the building blocks of a randomized subspace proximal gradient method *when the subspaces are independent and identically distributed.* Hence, we will leave out the possibility to adapt to the uncovered structure for now.

5.1.1 A random subspace proximal gradient algorithm

Our goal is to construct a *random subspace* version of the proximal gradient algorithm

$$u_k = x_k - \gamma \nabla f(x_k) \tag{5.1a}$$
$$x_{k+1} = \operatorname{prox}_{\gamma g}(y_k) \tag{5.1b}$$

By random subspace, we mean that that we will draw a linear subspace S_k *independently from a fixed distribution* and project the update on it. To do so, one has to determine which variable will be projected. Three choices are apparently possible:

(1) x_k, *i.e.* projecting after the proximity operation;

(2) $\nabla f(x_k)$, *i.e.* projecting after the gradient;

(3) u_k, *i.e.* projecting after the gradient *step*.

Choice (1) has limited interest in the general case where the proximity operator is not separable along subspaces and thus a projected update of x_k still requires the computations of the full gradient. In the favorable case of coordinate projection and $g = \|\cdot\|_1$, it was studied in (Qu and Richtárik, 2016) using the fact that the projection and the proximity operator commute. Choice (2) was considered recently in (Hanzely et al., 2018) in the slightly different context of sketching.

Here, we will consider Choice (3), inspired by recent works highlighting that combining iterates usually works well in practice (see *e.g.* (Mishchenko et al., 2020) and references therein). With this choice, a direct random projection such as

$$\begin{cases} y_k = \mathrm{proj}_{S_k}(x_k - \gamma \nabla f(x_k)) \\ x_{k+1} = \mathrm{prox}_{\gamma g}(y_k) \end{cases}$$

will in general bias the update and jeopardize convergence.

To overcome this issue, we include a correction using the inverse square root of the expected projection: $\mathrm{Q} := (\mathbb{E}[\mathrm{proj}_{S_k}])^{-1/2}$ which we assume to exist for now.[32]

Formally, our Random Proximal Subspace Descent (RPSD) algorithm, displayed as Algorithm 5.1, replaces (5.1a) by

$$z_k = \mathrm{Q}\left(x_k - \gamma \nabla f\left(x_k\right)\right) \qquad \text{and} \qquad y_k = \mathrm{proj}_{S_k}\left(z_k\right) + \left(\mathrm{I} - \mathrm{proj}_{S_k}\right)\left(y_{k-1}\right).$$

Intuitively, we first perform a gradient step followed by a change of basis (by multiplication with the positive definite matrix Q), giving variable z_k; then, variable y_k is updated only in the random subspace S_k, keeping the same value outside. Note that z_k does not actually have to be computed as only $\mathrm{proj}_{S_k} \mathrm{Q} \nabla f\left(x_k\right)$ is needed. Finally, the final proximal operation (5.1b) is performed after getting back to the original space (by multiplication with Q^{-1}):

$$x_{k+1} = \mathrm{prox}_{\gamma g}\left(\mathrm{Q}^{-1}\left(y_k\right)\right).$$

Contrary to existing coordinate descent methods, our randomized subspace proximal gradient algorithm does not assume that the proximity operator $\mathrm{prox}_{\gamma g}$ is separable with respect to the projection subspaces. Apart from the algorithm of (Hanzely et al., 2018) in a different setting, this is an uncommon but highly desirable feature to tackle general composite optimization problems.

[32] This is the main technical difficulty that limits us to linear subspaces. Indeed, in the case of linear subspaces, proj_{S_k} is a linear operator and its expectation is computable in many cases of interest as we will see in a short moment.

Algorithm 5.1 Randomized Proximal Subspace Descent - RPSD

Initialize $\mathrm{Q} = (\mathbb{E}[\mathrm{proj}_{S_0}])^{-1/2}$, y_0, $x_1 = \mathrm{prox}_{\gamma g}(\mathrm{Q}^{-1}(y_0))$
for $k = 1, \ldots$ **do**
  $z_k = \mathrm{Q}\left(x_k - \gamma \nabla f\left(x_k\right)\right)$
  $y_k = \mathrm{proj}_{S_k}\left(z_k\right) + \left(\mathrm{I} - \mathrm{proj}_{S_k}\right)\left(y_{k-1}\right)$
  $x_{k+1} = \mathrm{prox}_{\gamma g}\left(\mathrm{Q}^{-1}\left(y_k\right)\right)$
end for

Now that we have an algorithm in mind, let us investigate the choice of the random subspaces (S_k).

5.1.2 Subspace selection

We begin by introducing the mathematical objects leading to the subspace selection used in our randomized subspace descent algorithms. Though, in practice, most

algorithms rely on projection matrices, our presentation highlights intrinsic subspaces associated to these matrices; this opens the way to a finer analysis, especially in Section 5.2.2 when working with adaptive subspaces.

We consider a family $\mathsf{D} = \{\mathcal{S}_i\}_{i=1,..,p}$ of (linear) subspaces of $\mathbb{R}^n$. Intuitively, this set represents the directions that will be *favored* by the random descent; in order to reach a global optimum, we naturally assume that the sum of the subspaces in a family matches the whole space.[33]

[33] In the definition and the following, we use the natural set addition (sometimes called the Minkowski sum): for any two sets $\mathcal{C}, \mathcal{D} \subseteq \mathbb{R}^n$, the set $\mathcal{C} + \mathcal{D}$ is defined as $\{x + y : x \in \mathcal{C}, y \in \mathcal{D}\} \subseteq \mathbb{R}^n$.

Definition 5.2 (Covering family)**.** Let $\mathsf{D} = \{\mathcal{S}_i\}_{i=1,..,p}$ be a family of subspaces of $\mathbb{R}^n$. We say that D is *covering* if it spans the whole space, *i.e.* if $\sum_{i=1}^{p} \mathcal{S}_i = \mathbb{R}^n$.

Example 5.3. The family of the axes $\mathcal{S}_i = \{x \in \mathbb{R}^n : x^{[j]} = 0 \ \forall j \neq i\}$ for $i = 1, .., n$ is a canonical covering family for $\mathbb{R}^n$. ◀

From a covering family D, we call *selection* the random subspace obtained by randomly choosing some subspaces in D and summing them. We call *admissible* the selections that include all directions with some positive probability; or, equivalently, the selections to which no non-zero element of $\mathbb{R}^n$ is orthogonal with probability one.

Definition 5.4 (Admissible selection)**.** Let D be a covering family of subspaces of $\mathbb{R}^n$. A selection $\mathcal{S}$ on D is defined from the set of all subsets of $\{1, .., p\}$ to the set of the subspaces of $\mathbb{R}^n$ as

$$\mathcal{S}(\omega) = \sum_{j \in \omega} \mathcal{S}_j \qquad \text{for } \omega = \{i_1, \ldots, i_s\}.$$

The selection $\mathcal{S}$ is *admissible* if $\mathbb{P}[x \in \mathcal{S}^\perp] < 1$ for all $x \in \mathbb{R}^n \setminus \{0\}$.

Lemma 5.5 (Average projection)**.** *If a selection $\mathcal{S}$ is admissible then*

$$\mathsf{P} := \mathbb{E}[\mathrm{proj}_{\mathcal{S}}] \qquad \textit{is a positive definite matrix.}$$

In this case, we denote by $\lambda_{\min}(\mathsf{P}) > 0$ and $\lambda_{\max}(\mathsf{P}) \leq 1$ its minimal and maximal eigenvalues.

Proof. Note first that for almost all ω, the orthogonal projection $\mathrm{proj}_{\mathcal{S}(\omega)}$ is positive semi-definite, and therefore so is P. Now, let us prove that if P is not positive definite, then $\mathcal{S}$ is not admissible. Take a nonzero x in the kernel of P, then

$$x^\top \mathsf{P} x = 0 \iff x^\top \mathbb{E}[\mathrm{proj}_{\mathcal{S}}] x = 0 \iff \mathbb{E}[x^\top \mathrm{proj}_{\mathcal{S}}\, x] = 0.$$

Since $x^\top \mathrm{proj}_{\mathcal{S}(\omega)}\, x \geq 0$ for almost all ω, the above property is further equivalent for almost all ω to

$$x^\top \mathrm{proj}_{\mathcal{S}(\omega)}\, x = 0 \iff \mathrm{proj}_{\mathcal{S}(\omega)}\, x = 0 \iff x \in \mathcal{S}(\omega)^\perp.$$

As $x \neq 0$, this yields that $x \in \mathcal{S}(\omega)^\perp$ for almost all ω which is in contradiction with $\mathcal{S}$ being admissible. Thus, if a selection $\mathcal{S}$ is admissible, $\mathsf{P} := \mathbb{E}[\mathrm{proj}_{\mathcal{S}}]$ is positive definite (so $\lambda_{\min}(\mathsf{P}) > 0$).

Finally, using Jensen's inequality and the fact that $\mathrm{proj}_{\mathcal{S}}$ is a projection, we get $\|\mathsf{P}x\| = \|\mathbb{E}[\mathrm{proj}_{\mathcal{S}}]x\| \leq \mathbb{E}\|\mathrm{proj}_{\mathcal{S}}\, x\| \leq \|x\|$, which implies that $\lambda_{\max}(\mathsf{P}) \leq 1$. □

Although the framework, methods, and results presented in this paper allow for infinite subspace families (as in sketching algorithms); the most direct applications of our results only call for finite families for which the notion of admissibility can be simplified to $\mathbb{P}[\mathcal{S}_i \subset \mathcal{S}] > 0$ for all i.

5.1.3 Convergence and rate of RPSD

Now, we are in position to show that the proposed algorithm converges linearly at a rate that only depends on the function properties and on the smallest eigenvalue of P. We also emphasize that the step size γ can be taken in the usual range for the proximal gradient descent.

Theorem 5.6 (RPSD convergence rate). *Let Assumption 5.1 hold and let (S_k) be an i.i.d. sequence of admissible selections on the covering family* D. *Then, for any $\gamma \in (0, 2/(\mu+L)]$, the sequence (x_k) of the iterates of RPSD converges almost surely to the minimizer $x^\star$ of $(\mathcal{P}_A)$ with*

$$\mathbb{E}\left[\|x_{k+1} - x^\star\|_2^2\right] \leq \left(1 - \lambda_{\min}(\mathrm{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \lambda_{\max}(\mathrm{P})\|y_0 - \mathrm{Q}(x^\star - \gamma\nabla f(x^\star))\|_2^2.$$

To show this result, we first prove to "descent" lemmas that will capture the behavior of the method. One of the originalities of the proof is that some while the (x_k) are seen in the standard Euclidean way, the (z_k) are considered in the P-weighted norm $\|x\|_\mathrm{P} = \sqrt{\langle x, \mathrm{P}x\rangle}$ (and the (y_k) in both!).

Lemma 5.7 (Expression of the decrease as a martingale). *Let Assumption 5.1 hold and let (S_k) be an i.i.d. sequence of admissible selections on the covering family* D. *From the minimizer $x^\star$ of $(\mathcal{P}_A)$, define the fixed points $y^\star = z^\star = \mathrm{Q}\left(x^\star - \gamma\nabla f\left(x^\star\right)\right)$ of the sequences (z_k) and (y_k). Then*

$$\mathbb{E}\left[\|y_k - y^\star\|_2^2 | \mathcal{F}_{k-1}\right] = \|y_{k-1} - y^\star\|_2^2 + \|z_k - z^\star\|_\mathrm{P}^2 - \|y_{k-1} - y^\star\|_\mathrm{P}^2.$$

Proof. By taking the expectation on S_k (conditionally to the past), we get

$$\begin{aligned}
&\mathbb{E}\left[\|y_k - y^\star\|_2^2 | \mathcal{F}_{k-1}\right] \\
&= \mathbb{E}\left[\|y_{k-1} - y^\star + \mathrm{proj}_{S_k}(z_k - y_{k-1})\|_2^2 | \mathcal{F}_{k-1}\right] \\
&= \|y_{k-1} - y^\star\|_2^2 + 2\mathbb{E}\left[\langle y_{k-1} - y^\star, \mathrm{proj}_{S_k}(z_k - y_{k-1})\rangle | \mathcal{F}_{k-1}\right] + \mathbb{E}\left[\left\|\mathrm{proj}_{S_k}(z_k - y_{k-1})\right\|^2 | \mathcal{F}_{k-1}\right] \\
&= \|y_{k-1} - y^\star\|_2^2 + 2\langle y_{k-1} - y^\star, \mathrm{P}(z_k - y_{k-1})\rangle + \mathbb{E}\left[\langle \mathrm{proj}_{S_k}(z_k - y_{k-1}), \mathrm{proj}_{S_k}(z_k - y_{k-1})\rangle | \mathcal{F}_{k-1}\right] \\
&= \|y_{k-1} - y^\star\|_2^2 + 2\langle y_{k-1} - y^\star, \mathrm{P}(z_k - y_{k-1})\rangle + \mathbb{E}\left[\langle z_k - y_{k-1}, \mathrm{proj}_{S_k}(z_k - y_{k-1})\rangle | \mathcal{F}_{k-1}\right] \\
&= \|y_{k-1} - y^\star\|_2^2 + \langle y_{k-1} + z_k - 2y^\star, \mathrm{P}(z_k - y_{k-1})\rangle,
\end{aligned}$$

where we used the fact that y_{k-1} and z_k are $\mathcal{F}_{k-1}$-measurable and that proj_{S_k} is a projection matrix so $\mathrm{proj}_{S_k} = \mathrm{proj}_{S_k}^\top = \mathrm{proj}_{S_k}^2$.

Then, using the fact $z^\star = y^\star$, the scalar product above can be simplified as follows

$$\begin{aligned}
\langle y_{k-1} + z_k - 2y^\star, \mathrm{P}(z_k - y_{k-1})\rangle &= \langle y_{k-1} + z_k - y^\star - z^\star, \mathrm{P}(z_k - y_{k-1} + z^\star - y^\star)\rangle \\
&= -\langle y_{k-1} - y^\star, \mathrm{P}(y_{k-1} - y^\star)\rangle + \langle y_{k-1} - y^\star, \mathrm{P}(z_k - z^\star)\rangle \\
&\quad + \langle z_k - z^\star, \mathrm{P}(z_k - z^\star)\rangle - \langle z_k - z^\star, \mathrm{P}(y_{k-1} - y^\star)\rangle \\
&= \langle z_k - z^\star, \mathrm{P}(z_k - z^\star)\rangle - \langle y_{k-1} - y^\star, \mathrm{P}(y_{k-1} - y^\star)\rangle
\end{aligned}$$

where we used in the last equality that P is symmetric. □

Lemma 5.8 (Contraction property in P-weighted norm). *Let Assumption 5.1 hold and let (S_k) be an i.i.d. sequence of admissible selections on the covering family* D. *From the minimizer $x^\star$ of $(\mathcal{P}_A)$, define the fixed points $y^\star = z^\star = \mathrm{Q}\left(x^\star - \gamma\nabla f\left(x^\star\right)\right)$ of the*

sequences (z_k) *and* (y_k)*. Then*

$$\|z_k - z^\star\|_{\mathsf{P}}^2 - \|y_{k-1} - y^\star\|_{\mathsf{P}}^2 \leq -\lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\|y_{k-1} - y^\star\|_2^2.$$

Proof. First, using the definition of z_k and $z^\star$,

$$\begin{aligned}\|z_k - z^\star\|_{\mathsf{P}}^2 &= \langle \mathsf{Q}(x_k - \gamma\nabla f(x_k) - x^\star + \gamma\nabla f(x^\star)), \mathsf{PQ}(x_k - \gamma\nabla f(x_k) - x^\star + \gamma\nabla f(x^\star))\rangle \\ &= \langle x_k - \gamma\nabla f(x_k) - x^\star + \gamma\nabla f(x^\star), \mathsf{Q}^\top\mathsf{PQ}(x_k - \gamma\nabla f(x_k) - x^\star + \gamma\nabla f(x^\star))\rangle \\ &= \left\|x_k - \gamma\nabla f(x_k) - (x^\star - \gamma\nabla f(x^\star))\right\|_2^2.\end{aligned}$$

Using the standard stepsize range $\gamma \in (0, 2/(\mu + L)]$, one has (see Section 2.3.1)

$$\|z_k - z^\star\|_{\mathsf{P}}^2 = \left\|x_k - \gamma\nabla f(x_k) - (x^\star - \gamma\nabla f(x^\star))\right\|_2^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)\|x_k - x^\star\|_2^2.$$

Using the non-expansiveness of the proximity operator of the convex lower semi-continuous function g along with the fact that as $x^\star$ is a minimizer of $(\mathcal{P}_{\mathsf{A}})$ so $x^\star = \operatorname{prox}_{\gamma g}(x^\star - \gamma\nabla f(x^\star)) = \operatorname{prox}_{\gamma g}(\mathsf{Q}^{-1}y^\star)$ (see Section 2.3.2), we get

$$\begin{aligned}\|x_k - x^\star\|_2^2 &= \|\operatorname{prox}_{\gamma g}(\mathsf{Q}^{-1}(y_{k-1})) - \operatorname{prox}_{\gamma g}(\mathsf{Q}^{-1}(y^\star))\|_2^2 \\ &\leq \|\mathsf{Q}^{-1}(y_{k-1} - y^\star)\|_2^2 = \langle \mathsf{Q}^{-1}(y_{k-1} - y^\star), \mathsf{Q}^{-1}(y_{k-1} - y^\star)\rangle \\ &= \langle y_{k-1} - y^\star, \mathsf{P}(y_{k-1} - y^\star)\rangle = \|y_{k-1} - y^\star\|_{\mathsf{P}}^2\end{aligned}$$

where we used that $\mathsf{Q}^{-\top}\mathsf{Q}^{-1} = \mathsf{Q}^{-2} = \mathsf{P}$. Combining the previous equations, we get

$$\|z_k - z^\star\|_{\mathsf{P}}^2 - \|y_{k-1} - y^\star\|_{\mathsf{P}}^2 \leq -\frac{2\gamma\mu L}{\mu + L}\|y_{k-1} - y^\star\|_{\mathsf{P}}^2.$$

Finally, the fact that $\|x\|_{\mathsf{P}}^2 \geq \lambda_{\min}(\mathsf{P})\|x\|_2^2$ for positive definite matrix P enables to get the claimed result. □

Relying on these two lemmas, we are now able to prove Theorem 5.6 by showing that the distance of y_k towards the minimizers is a contracting super-martingale.

Proof of Theorem 5.6. Combining Lemmas 5.7 and 5.8, we get

$$\mathbb{E}\left[\|y_k - y^\star\|_2^2 | \mathcal{F}_{k-1}\right] \leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)\|y_{k-1} - y^\star\|_2^2$$

and thus by taking the full expectation and using nested filtrations $(\mathcal{F}_k)$, we obtain

$$\begin{aligned}\mathbb{E}\left[\|y_k - y^\star\|_2^2\right] &\leq \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \|y_0 - y^\star\|_2^2 \\ &= \left(1 - \lambda_{\min}(\mathsf{P})\frac{2\gamma\mu L}{\mu + L}\right)^k \|y_0 - \mathsf{Q}(x^\star - \gamma\nabla f(x^\star))\|_2^2.\end{aligned}$$

Using the same arguments as in the proof of Lemma 5.8, one has

$$\|x_{k+1} - x^\star\|_2^2 \leq \|y_k - y^\star\|_{\mathsf{P}}^2 \leq \lambda_{\max}(\mathsf{P})\|y_k - y^\star\|_2^2$$

which enables to conclude

$$\mathbb{E}\left[\|x_{k+1}-x^\star\|_2^2\right] \leq \left(1-\lambda_{\min}(\mathrm{P})\frac{2\gamma\mu L}{\mu+L}\right)^k \lambda_{\max}(\mathrm{P})\|y_0-\mathrm{Q}(x^\star-\gamma\nabla f(x^\star))\|_2^2.$$

Finally, this linear convergences implies the almost sure convergence of (x_k) to $x^\star$ as

$$\mathbb{E}\left[\sum_{k=1}^{+\infty}\|x_{k+1}-x^\star\|^2\right] \leq C\sum_{k=1}^{+\infty}\left(1-\lambda_{\min}(\mathrm{P})\frac{2\gamma\mu L}{\mu+L}\right)^k < +\infty$$

implies that $\sum_{k=1}^{+\infty}\|x_{k+1}-x^\star\|^2$ is finite with probability one. Thus we get

$$1=\mathbb{P}\left[\sum_{k=1}^{+\infty}\|x_{k+1}-x^\star\|^2 < +\infty\right] \leq \mathbb{P}\left[\|x_k-x^\star\|^2\to 0\right]$$

which in turn implies that (x_k) converges almost surely to $x^\star$. □

5.1.4 *The sparse case*

A simple instantiation of our setting can be obtained by considering projections onto uniformly chosen coordinates, which enables to recover standard coordinate descent.

We choose the family

$$\mathrm{D}=\{\mathcal{S}_1,..,\mathcal{S}_n\} \quad \text{with } \mathcal{S}_i=\{x\in\mathbb{R}^n : x^{[j]}=0 \ \ \forall j\neq i\}$$

and the selection S consisting of taking $\mathcal{S}_i$ according to the output of an independent Bernoulli variable of parameter $p_i\in(0,1]$. Then, the matrices $\mathrm{P}=\mathrm{diag}([p_1,..,p_n])$, $\mathrm{proj}_{\mathcal{S}_k}$ and Q commute, and, by a change of variables $\tilde{y}_k=\mathrm{Q}^{-1}y_k$ and $\tilde{z}_k=\mathrm{Q}^{-1}z_k$, Algorithm 5.1 boils down to

$$\tilde{z}_k=x_k-\gamma\nabla f(x_k), \qquad \tilde{y}_k=\mathrm{proj}_{\mathcal{S}_k}(\tilde{z}_k)+(\mathrm{I}-\mathrm{proj}_{\mathcal{S}_k})(\tilde{y}_{k-1}), \qquad x_{k+1}=\mathrm{prox}_{\gamma g}(\tilde{y}_k)$$

i.e. no change of basis is needed anymore, even if g is non-separable. Furthermore, the convergence rates simplifies to $(1-2\min_i p_i\gamma\mu L/(\mu+L))$ which translates to $(1-4\min_i p_i\mu L/(\mu+L)^2)$ for the optimal $\gamma=2/(\mu+L)$.

In the special case where g is separable (*i.e.* $g(x)=\sum_{i=1}^n g_i(x^{[i]})$), we can further simplify the iteration. In this case, projection and proximal steps commute, so that the iteration can be written

$$x_{k+1}=\mathrm{proj}_{\mathcal{S}_k}\mathrm{prox}_{\gamma g}(x_k-\gamma\nabla f(x_k))+(\mathrm{I}-\mathrm{proj}_{\mathcal{S}_k})x_k, \text{ i.e.}$$

$$x_{\ell+1}^{[i]}=\begin{cases}\mathrm{prox}_{\gamma g_i}\left(x_k^{[i]}-\gamma\nabla_i f(x_k)\right)=\mathrm{argmin}_u\, g_i(u)+\langle u,\nabla_i f(x_k)\rangle+\frac{1}{2\gamma}\|u-x_k^{[i]}\|_2^2 & \text{if } i\in S_k\\ x_k^{[i]} & \text{elsewhere}\end{cases}$$

which boils down to the usual (proximal) coordinate descent algorithm, that recently knew a rebirth in the context of huge-scale optimization, see (Nesterov, 2012; Richtárik and Takáč, 2014; Tseng, 2001; Wright, 2015). In this special case, the theoretical convergence rate of RPSD is similar to the existing rates in the literature, see *e.g.* (Richtárik and Takáč, 2014).

5.1.5 Identification holds... but is inexploitable

Following Section 3.2.2, we investigate the identification properties of RPSD by defining a collection of manifolds:

$$\mathrm{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}$$

and recall our notation for structure as mapping $\mathrm{S} : \mathbb{R}^n \to \{0,1\}^q$ defined as

$$\mathrm{S}^{[i]}(x) = 0 \quad \text{if } x \in \mathcal{M}_i \text{ and } 1 \text{ elsewhere.}$$

Now, Theorem 5.6 guarantees that the iterates of RPSD converge almost surely to $x^\star$ which in turn implies that $\mathrm{Q}^{-1}y_k$ converges almost surely to $x^\star - \gamma\nabla f(x^\star)$. Thus, with probability one, Theorem 3.4 applies. This gives the following result.

Theorem 5.9 (Partial identification)**.** *Let Assumption 5.1 hold and let $(\mathcal{S}_k)$ be an i.i.d. sequence of admissible selections on the covering family* D*. Then, for any $\gamma \in (0, 2/(\mu{+}L)]$, with probability one, the iterates (x_k) of RPSD satisfy after some finite time*

$$\mathrm{S}^{[i]}(x^\star) \;\le\; \mathrm{S}^{[i]}(x_k) \;\le\; \max\left\{\mathrm{S}^{[i]}\big(\mathrm{prox}_{\gamma g}(y)\big) : \|y - x^\star + \gamma\nabla f(x^\star)\| \le \varepsilon\right\}$$

for all $i = 1, .., q$.

Thus, we indeed have identification but it is unexploitable since our subspace selection is i.i.d.. Following Section 3.5, we would like to favor the directions along the identified manifolds. Intuitively, this means going from the i.i.d. choice of RPSD:

$$\mathcal{S}_i \in \mathcal{S}_k \text{ with probability } p_i \text{ for all } k, i$$

to an adaptive sampling based on the currently identified manifolds:

$$\mathcal{S}_i \in \mathcal{S}_k \text{ with probability } \begin{cases} p & \text{if } x_k \in \mathcal{M}_i \text{ and } \mathcal{S}_i \text{ is orthogonal to } \mathcal{M}_i \\ 1 & \text{elsewhere} \end{cases} \quad \text{for all } i.$$

This adaptive strategy means always updating in the direction (tangent to) the identified manifolds;[34] but only with some probability $p > 0$ orthogonally to them (in order not to be stuck in a manifold and to retain the convergence guarantees).

[34] Recall that the $\mathcal{S}_i$, and in fact the $\mathcal{M}_i$ are all linear subspaces of $\mathbb{R}^n$ in this chapter.

We develop this idea along with the necessary mathematical tools in the next section.

5.2 Adaptive subspace descent

Our aim in this section is to automatically adapt to the structure identified by the iterates along the run of the algorithm. The methods proposed here are, up to our knowledge, the first ones where the iterate structure enforced by a nonsmooth regularizer is used to adapt the selection probabilities in a randomized first-order method.

5.2.1 Structure & Subspace

We first provide some general rules to link the sampling (thus the family of subspaces D) with the identified structure (the family of possible manifolds C). To this end, the two families D and C have to be closely related. We thus introduce the notion of generalized complemented subspaces.

Definition 5.10 (Generalized complemented subspaces)**.** Two families of subspaces $\mathsf{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}$ and $\mathsf{D} = \{\mathcal{S}_1, \dots, \mathcal{S}_q\}$ are said to be (generalized) complemented subspaces if for all $i = 1, \dots, q$

$$\begin{cases} (\mathcal{S}_i \bigcap \mathcal{M}_i) \subseteq \bigcap_j \mathcal{S}_j \\ \mathcal{S}_i + \mathcal{M}_i = \mathbb{R}^n \end{cases}.$$

Example 5.11 (Complemented subspaces and sparsity vectors). For sparsity patterns, the structure can be captured by the collection (see Section 3.2.3)

$$\mathsf{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_n\} \quad \text{with } \mathcal{M}_i = \{x \in \mathbb{R}^n : x^{[i]} = 0\},$$

then, a natural choice for the subspace is the collection of the axes (see Example 5.3)

$$\mathsf{D} = \{\mathcal{S}_1, \dots, \mathcal{S}_n\} \qquad \text{with } \mathcal{S}_i = \{x \in \mathbb{R}^n : x^{[j]} = 0 \ \ \forall j \neq i\}.$$

They are complemented subspaces since $\mathcal{M}_i \bigcap \mathcal{S}_i = \{0\} = \bigcap_j \mathcal{S}_j$ and $\mathcal{S}_i + \mathcal{M}_i = \mathbb{R}^n$. In this case, the sparsity vector $\mathsf{S}(x)$ corresponds to the *support* of x (indeed $\mathsf{S}^{[i]}(x) = 0$ if and only if $x \in \mathcal{M}_i \Leftrightarrow x^{[i]} = 0$). ◀

The practical reasoning with using complemented families is the following. If the subspace $\mathcal{M}_i$ is definitively identified at time K (*i.e.* $\mathsf{S}^{[i]}(x_k) = 0 \Leftrightarrow x_k \in \mathcal{M}_i$ for all $k \geq K$), then it is no use to update the iterates in $\mathcal{S}_i$ in preference, and the next selection S_k should not include $\mathcal{S}_i$ anymore. Unfortunately, the moment after which a subspace is definitively identified is unknown in general; however, subspaces $\mathcal{M}_i$ usually show a certain stability and thus $\mathcal{S}_i$ may be "less included" in the selection. This is the intuition behind our adaptive subspace descent algorithm. Once again, this reasoning is numerically tractable since the computation of the proximity operator comes with the knowledge of the structure of the output; see Sections 3.2.3 and 3.5.

5.2.2 Adaptive Random Subspace Descent

For any randomized algorithm, using iterate-dependent sampling automatically breaks down the i.i.d. assumption. In our case, adapting to the current iterate structure means that the associated random variable depends on the past. We thus need further analysis and notation.

In the following, we use the subscript ℓ to denote the ℓ-th change in the selection. We denote by L the set of time indices at which an adaptation is made, themselves denoted by $k_\ell = \min\{k > k_{\ell-1} : k \in \mathsf{L}\}$. We then replace the i.i.d. assumption by the following one.

Assumption 5.12 (On the randomness of the adaptive algorithm)**.** For all $k > 0$, S_k is $\mathcal{F}_k$-measurable and admissible. Furthermore, for any ℓ, $k \geq k_\ell$, (S_k) is independent and identically distributed on $[k_\ell, k]$. The decision to adapt or not at time k is $\mathcal{F}_k$-measurable, *i.e.* $(k_\ell)_\ell$ is a sequence of $\mathcal{F}_k$-stopping times.

This means that the subspace sampling distribution can be changed at some iterations of the algorithm, the ones in L, but between them, the sampling is i.i.d.. This gives the generic Algorithm 5.2, where we denote by $\mathsf{P}_\ell = \mathbb{E}[\mathrm{proj}_{S_{k_\ell}}]$ the average projection for the i.i.d. sequence $(S_{k_\ell}, S_{k_\ell+1}, .., S_{k_{\ell+1}-1})$, and $\mathsf{Q}_\ell = \mathsf{P}_\ell^{-\frac{1}{2}}$ accordingly.

Under this assumption, we can prove the convergence of this random subspace descent method with time-varying selections. The rationale of the proof is that the

stability of the algorithm is maintained when adaptation is performed sparingly. After analyzing this generic algorithm, we will see how this can be put into practice.

Algorithm 5.2 Adaptive Randomized Proximal Subspace Descent - ARPSD

Initialize $\mathsf{Q} = (\mathbb{E}[\mathrm{proj}_{S_0}])^{-1/2}$, $y_0, x_1 = \mathrm{prox}_{\gamma g}(\mathsf{Q}_0^{-1}(y_0))$, $\ell = 0$, $\mathsf{L} = \{0\}$.
for $k = 1, \dots$ **do**
  $z_k = \mathsf{Q}_\ell \left(x_k - \gamma \nabla f\left(x_k\right)\right)$
  $y_k = \mathrm{proj}_{S_k}\left(z_k\right) + \left(\mathrm{I} - \mathrm{proj}_{S_k}\right)\left(y_{k-1}\right)$
  $x_{k+1} = \mathrm{prox}_{\gamma g}\left(\mathsf{Q}_\ell^{-1}\left(y_k\right)\right)$
  if an adaptation is decided **then**
    $\mathsf{L} \leftarrow \mathsf{L} \cup \{k+1\}$, $\ell \leftarrow \ell + 1$
    Generate a new admissible selection with $\mathsf{P}_\ell = \mathbb{E}[\mathrm{proj}_{S_{k_\ell}}]$
    Compute $\mathsf{Q}_\ell = \mathsf{P}_\ell^{-\frac{1}{2}}$ and Q_ℓ^{-1}
    Rescale $y_k \leftarrow \mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1} y_k$
  end if
end for

Before going further on the applicability of this method with respect to identification and adapted sampling strategies, let us see what theoretical results can be obtained.

5.2.3 Convergence and rate

Under Assumption 5.12, the convergence is obviously not unconditional: all the sampling strategies have to be admissible and cannot change too much and too often. The "crude" result we can obtain is the following.

Theorem 5.13 (ARPSD convergence)**.** *Let Assumption 5.1 and Assumption 5.12 hold. For any $\gamma \in (0, 2/(\mu + L)]$, let the user choose its adaptation strategy so that:*

i) the adaptation cost *is upper bounded deterministically: $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2 \leq \mathrm{a}_\ell$;*

ii) the inter-adaptation time *is lower bounded deterministically: $k_\ell - k_{\ell-1} \geq \mathrm{c}_\ell$;*

iii) the selection admissibility *is lower bounded deterministically: $\lambda_{\min}(\mathsf{P}_{\ell-1}) \geq \lambda_{\ell-1}$;*

then, from the previous instantaneous rate $1 - \alpha_{\ell-1} := 1 - 2\gamma\mu L\lambda_{\ell-1}/(\mu + L)$*, the* corrected rate *for cycle ℓ writes*

$$(1 - \beta_k) := (1 - \alpha_{\ell-1})\mathrm{a}_\ell^{1/\mathrm{c}_\ell}. \tag{5.2}$$

Then, we have for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|x_{k+1} - x^\star\|_2^2\right] \leq (1 - \alpha_\ell)^{k-k_\ell} \prod_{m=1}^{\ell} (1 - \beta_m)^{\mathrm{c}_m} \|y_0 - \mathsf{Q}_0\left(x^\star - \gamma \nabla f\left(x^\star\right)\right)\|_2^2.$$

This theorem means that by balancing the magnitude of the adaptation (*i.e.* a_ℓ) with the time before adaptation (*i.e.* c_ℓ) from the knowledge of the current rate $(1-\alpha_{\ell-1})$, one can retrieve the exponential convergence with a controlled degraded rate $(1-\beta_\ell)$. This result is quite generic, but it can be easily adapted to specific situations. For instance, we provide a simple example with a global rate on the iterates in the forthcoming Example 5.14.

For now, let us turn to the proof of the theorem. To ease its reading, the main notations and measurability relations are depicted in Fig. 5.1.

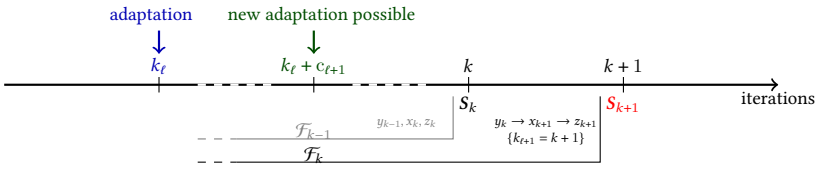


Figure 5.1: Summary of notations about iteration, adaptation and filtration. The filtration $\mathcal{F}_{k-1}$ is the sigma-algebra generated by $\{S_{k'}\}_{k' \leq k-1}$ encompassing the knowledge of all variables up to z_k (but not y_k).

Proof. Let us define $y_\ell^\star = \mathrm{Q}_\ell \left(x^\star - \gamma \nabla f\left(x^\star\right)\right)$, Lemmas 5.7 and 5.8 can be directly extended and combined to show for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|y_k - y_\ell^\star\|_2^2 | \mathcal{F}_{k-1}\right] \leq \underbrace{\left(1 - \frac{2\gamma\mu L \lambda_{\min}(\mathrm{P}_\ell)}{\mu + L}\right)}_{\leq 1 - \alpha_\ell} \|y_{k-1} - y_\ell^\star\|_2^2. \tag{5.3}$$

Since the distribution of the selection has not changed since k_ℓ, iterating (5.3) leads to

$$\mathbb{E}\left[\|y_k - y_\ell^\star\|_2^2 | \mathcal{F}_{k_\ell - 1}\right] \leq (1 - \alpha_\ell)^{k - k_\ell} \|y_{k_\ell - 1} - y_\ell^\star\|_2^2. \tag{5.4}$$

We focus now on the term $\|y_{k_\ell - 1} - y_\ell^\star\|_2^2$ corresponding to what happens at the last adaptation step. From the definition of variables in the algorithm and using the deterministic bound on $\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|$, we write

$$\begin{aligned}
&\mathbb{E}\left[\|y_{k_\ell - 1} - y_\ell^\star\|_2^2 | \mathcal{F}_{k_\ell - 2}\right] \\
&\leq \mathbb{E}\left[\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}(y_{k_\ell - 2} + \mathrm{proj}_{S_{k_\ell - 1}}(z_{k_\ell - 1} - y_{k_\ell - 2}) - \mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1} y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_\ell - 2}\right] \\
&\leq \mathbb{E}\left[\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2 \|y_{k_\ell - 2} + \mathrm{proj}_{S_{k_\ell - 1}}(z_{k_\ell - 1} - y_{k_\ell - 2}) - y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_\ell - 2}\right] \\
&\leq \mathrm{a}_\ell (1 - \alpha_{\ell-1}) \|y_{k_\ell - 2} - y_{\ell-1}^\star\|_2^2.
\end{aligned} \tag{5.5}$$

Repeating this inequality backwards to the previous adaptation step $y_{k_{\ell-1}}$, we get

$$\begin{aligned}
\mathbb{E}\left[\|y_{k_\ell - 1} - y_\ell^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right] &\leq \mathrm{a}_\ell (1 - \alpha_{\ell-1})^{k_\ell - k_{\ell-1}} \|y_{k_{\ell-1}} - y_{\ell-1}^\star\|_2^2 \\
&\leq \mathrm{a}_\ell (1 - \alpha_{\ell-1})^{\mathrm{c}_\ell} \|y_{k_{\ell-1}} - y_{\ell-1}^\star\|_2^2,
\end{aligned}$$

using the assumption of bounded inter-adaptation times. Combining this inequality and (5.4), we obtain that for any $k \in [k_\ell, k_{\ell+1})$,

$$\mathbb{E}\left[\|y_k - y_\ell^\star\|_2^2\right] \leq (1 - \alpha_\ell)^{k - k_\ell} \prod_{m=1}^{\ell} \mathrm{a}_m (1 - \alpha_{m-1})^{\mathrm{c}_m} \|y_0 - y_0^\star\|_2^2.$$

Using now (5.2), we get

$$\mathbb{E}\left[\|y_k - y_\ell^\star\|_2^2\right] \leq (1 - \alpha_\ell)^{k - k_\ell} \prod_{m=1}^{\ell} (1 - \beta_m)^{\mathrm{c}_m} \|y_0 - y_0^\star\|_2^2$$

Finally, the non-expansiveness of the prox-operator propagates this inequality to x_k, since we have

$$\begin{aligned}\|x_k - x^\star\|_2^2 &= \|\mathrm{prox}_{\gamma g}(\mathrm{Q}_\ell^{-1}(y_{k-1})) - \mathrm{prox}_{\gamma g}(\mathrm{Q}_\ell^{-1}(y_\ell^\star))\|_2^2 \\ &\leq \|\mathrm{Q}_\ell^{-1}(y_{k-1} - y_\ell^\star)\|_2^2 \leq \lambda_{\max}(\mathrm{Q}_\ell^{-1})^2 \|y_{k-1} - y_\ell^\star\|_2^2 \\ &= \lambda_{\max}(\mathrm{P}_\ell)\|y_{k-1} - y_\ell^\star\|_2^2 \leq \|y_{k-1} - y_\ell^\star\|_2^2.\end{aligned}$$

This concludes the proof. □

Example 5.14 (Explicit convergence rate). Let us specify Theorem 5.13 with the following simple adaptation strategy. We take a fixed upper bound on the adaptation cost and a fixed lower bound on uniformity:

$$\|\mathrm{Q}_k \mathrm{Q}_{k-1}^{-1}\|_2^2 \leq \mathrm{a} \qquad \lambda_{\min}(\mathrm{P}_k) \geq \lambda. \tag{5.6}$$

Then from the rate $1 - \alpha = 1 - 2\gamma\mu L\lambda/(\mu + L)$, we can perform an adaptation every

$$\mathrm{c} = \lceil \log(\mathrm{a}) / \log\left((2-\alpha)/(2-2\alpha)\right) \rceil \tag{5.7}$$

iterations, so that $\mathrm{a}(1-\alpha)^{\mathrm{c}} = (1-\alpha/2)^{\mathrm{c}}$ and $k_\ell = k\mathrm{c}$. A direct application of Theorem 5.13 gives that, for any k,

$$\mathbb{E}\left[\|x_{k+1} - x^\star\|_2^2\right] \leq \left(1 - \frac{\gamma\mu L\lambda}{\mu + L}\right)^k C$$

where $C = \|y_0 - \mathrm{Q}_0(x^\star - \gamma\nabla f(x^\star))\|_2^2$. That is the same convergence mode as in the non-adaptive case (Theorem 5.6) with a modified rate. Note the modified rate provided here (of the form $(1-\alpha/2)$ to be compared with the $1-\alpha$ of Theorem 5.6) was chosen for clarity; any rate strictly slower than $1-\alpha$ can bring the same result by adapting c accordingly. ◀

Remark 5.15 (On the adaptation frequency). Theorem 5.13 and Example 5.14 tell us that we have to respect a prescribed number of iterations between two adaptation steps. We emphasize here that if this inter-adaptation time is violated, the resulting algorithm may be highly unstable. We illustrate this phenomenon on a TV-regularized least squares problem: we compare two versions of ARPSD with the same adaptation strategy verifying (5.6) but with two different adaptation frequencies

- at every iteration (*i.e.* taking $\mathrm{c}_\ell = 1$)
- following theory (*i.e.* taking $\mathrm{c}_\ell = \mathrm{c}$ as per Eq. (5.7))

On Fig. 5.2, we observe that adapting every iteration leads to a chaotic behavior. Second, even though the theoretical number of iterations in an adaptation cycle is often pessimistic (due to the rough bounding of the rate), the iterates produced with this choice quickly become stable (*i.e.* identification happens, which will be shown and exploited in the next section) and show a steady decrease in suboptimality. ◀

A drawback of Theorem 5.13 is that the adaptation cost, inter-adaptation time, and selection uniformity have to be bounded by deterministic sequences. This can be restrictive if we do not have prior knowledge on the problem or if the adaptation cost varies a lot. This drawback can be circumvented to the price of loosing the rate *per iteration* to the rate *per adaptation*, as formalized in the following result.

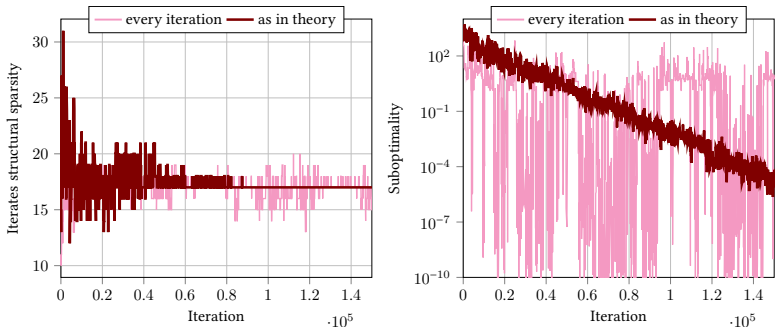


Figure 5.2: Comparisons between theoretical and harsh updating time for ARPSD.

Theorem 5.16 (ARPSD convergence: practical version). *Let Assumptions 5.1 and 5.12 hold. Take $\gamma \in (0, 2/(\mu + L)]$, choose $\lambda > 0$, and set $\beta = \gamma\mu L\lambda/(\mu + L)$. Consider the following adaptation strategy:*

1) *From the observation of $x_{k_{\ell-1}}$ choose a new sampling with P_ℓ and Q_ℓ, such that $\lambda_{\min}(\mathrm{P}_\ell) \geq \lambda$;*
2) *Compute c_ℓ so that $\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2 (1 - \alpha_{\ell-1})^{\mathrm{c}_\ell} \leq 1 - \beta$ where $\alpha_{\ell-1} = 2\gamma\mu L\lambda_{\min}(\mathrm{P}_{\ell-1})/(\mu + L)$;*
3) *Apply the new sampling after c_ℓ iterations ($k_\ell = k_{\ell-1} + \mathrm{c}_\ell$).*

Then, we have for any $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left[\|x_{k+1} - x^\star\|_2^2\right] \leq (1 - \alpha_\ell)^{k-k_\ell} (1 - \beta)^\ell \|y_0 - \mathrm{Q}_0 \left(x^\star - \gamma \nabla f\left(x^\star\right)\right)\|_2^2.$$

Proof. The proof follows the same pattern as the one of Theorem 5.13. The only difference is that the three control sequences (adaptation cost, inter-adaptation time, and selection uniformity) are now random sequences since they depend on the iterates of the (random) algorithm. This technical point requires a special attention. In (5.5), the adaptation introduces a cost by a factor $\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2$, which is not deterministically upper-bounded anymore. However it is $\mathcal{F}_{k_{\ell-1}}$-measurable by construction of Q_ℓ, so we can write

$$\begin{aligned}
&\mathbb{E}\left[\|y_{k_\ell-1} - y_\ell^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\|y_{k_\ell-1} - y_\ell^\star\|_2^2 | \mathcal{F}_{k_\ell-2}\right] | \mathcal{F}_{k_{\ell-1}}\right] \\
&\leq \mathbb{E}\left[\mathbb{E}\left[\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}(y_{k_\ell-2} + \mathrm{proj}_{S_{k_{\ell-1}}}(z_{k_\ell-1} - y_{k_\ell-2}) - \mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1} y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_\ell-2}\right] | \mathcal{F}_{k_{\ell-1}}\right] \\
&\leq \mathbb{E}\left[\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2 (1 - \alpha_{\ell-1}) \|y_{k_\ell-2} - y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right] \\
&= \|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2 (1 - \alpha_{\ell-1}) \mathbb{E}\left[\|y_{k_\ell-2} - y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right].
\end{aligned}$$

Using Eq. (5.3), this inequality yields

$$\begin{aligned}
\mathbb{E}\left[\|y_{k_\ell-1} - y_\ell^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right] &\leq \|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2 (1 - \alpha_{\ell-1})^{k_\ell - k_{\ell-1}} \mathbb{E}\left[\|y_{k_{\ell-1}-1} - y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right] \\
&\leq (1 - \beta) \mathbb{E}\left[\|y_{k_{\ell-1}-1} - y_{\ell-1}^\star\|_2^2 | \mathcal{F}_{k_{\ell-1}}\right].
\end{aligned}$$

where we used points 2) and 3) of the strategy to bound the first terms deterministically. Finally, we obtain

$$\begin{aligned}\mathbb{E}\left[\|y_{k_\ell-1}-y_\ell^\star\|_2^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\|y_{k_\ell-1}-y_\ell^\star\|_2^2|\mathcal{F}_{k_{\ell-1}}\right]\right]\\ &\leq (1-\beta)\mathbb{E}\left[\|y_{k_{\ell-1}-1}-y_{\ell-1}^\star\|_2^2\right]\end{aligned}$$

then the rest of the proof follows directly by induction. □

5.2.4 *Adaptivity in Practice & Comparison between RPSD and ARPSD*

Now that we know to what extent we can change sampling strategies, we can implement our structure adapted strategies. We recall that we can identify structure among a family of linear subspaces $\mathsf{C} = \{\mathcal{M}_1, \dots, \mathcal{M}_q\}$ and that our sampling directions $\mathsf{D} = \{\mathcal{S}_1, \dots, \mathcal{S}_q\}$ are complemented subspaces to C.

Intuitively, if $\mathcal{M}_i$ is identified (*i.e.* $x_k \in \mathcal{M}_i$), then $\mathcal{S}_i$, which is orthogonal to $\mathcal{M}_i$ has less interest. So we propose to construct S_k such that $\mathbb{P}[\mathcal{S}_i \in S_k] = p$ for a small probability $p > 0$ for all i such that $\mathcal{M}_i$ is identified; and $\mathbb{P}[\mathcal{S}_i \in S_k] = 1$ for the others.

Unfortunately, we saw before that adapting at every iteration was impossible. Hence, at time $k \in [k_\ell, k_{\ell+1})$, the structure that is used for the sampling is the one of time $k_{\ell-1}$ (as per Theorem 5.16). Table 5.1 summarizes the common points and differences between the proposed adaptive and non-adaptive subspace descent methods.

		(non-adaptive) subspace descent RPSD	adaptive subspace descent ARPSD
Subspace family		$\mathsf{D} = \{\mathcal{S}_1, .., \mathcal{S}_c\}$	
Algorithm		$\begin{cases} z_k = & \mathrm{Q}\left(x_k - \gamma\nabla f\left(x_k\right)\right) \\ y_k = & \mathrm{proj}_{S_k}\left(z_k\right) + \left(\mathrm{I} - \mathrm{proj}_{S_k}\right)\left(y_{k-1}\right) \\ x_{k+1} = & \mathrm{prox}_{\gamma g}\left(\mathrm{Q}^{-1}\left(y_k\right)\right) \end{cases}$	
Selection	Option 1	$\mathcal{S}_i \in S_k$ with probability p	$\mathcal{S}_i \in S_k$ with probability $\begin{cases} p & \text{if } x_{k_{\ell-1}} \in \mathcal{M}_i \\ 1 & \text{elsewhere} \end{cases}$
	Option 2	Sample s elements uniformly in D	Sample s elements uniformly in $\{\mathcal{S}_i : x_{k_{\ell-1}} \in \mathcal{M}_i\}$ and add *all* elements in $\{\mathcal{S}_j : x_{k_{\ell-1}} \notin \mathcal{M}_j\}$

Table 5.1: Strategies for non-adaptive vs. adaptive algorithms. The two options introduced in this table are examples on how to generate reasonably performing admissible selections. Their difference lies in the fact that for Option 1, the *probability* of sampling a subspace outside the support is controled, while for Option 2, the *number* of subspaces is controlled (this makes every iteration computationally similar which can be interesting in practice). Option 2 will be discussed in Section 5.3 and illustrated numerically.

Notice that, contrary to the importance-like adaptive algorithms of (Stich et al., 2017) for instance, the purpose of these methods is not to adapt each subspace probability to local *steepness* but rather to adapt them to the current *structure*. This is notably due to the fact that local steepness-adapted probabilities can be difficult to evaluate numerically and that in heavily structured problems, adapting to an ultimately very sparse structure already reduces drastically the number of explored dimensions, as suggested in (Grishchenko et al., 2021) for the case of coordinate-wise projections.

5.2.5 Rate improvement with identification

Now, in addition to the expected computational advantage of adaptive sampling, we can actually have a better theoretical rate under the usual proximal gradient qualification constraint to guarantee exact identification:

$$\text{for all } y \in \mathcal{B}(\overline{x} - \gamma \nabla f(\overline{x}), \varepsilon), \quad \operatorname{prox}_{\gamma g}(y) \in \mathcal{M}, \qquad \text{(PQC – Proximal Gradient)}$$

Theorem 5.17 (Improved asymptotic rate). *Under the same assumptions as in Theorems 5.13 and 5.16, if the solution $x^\star$ of ($\mathcal{P}_A$) verifies the qualification constraint* (PQC – Proximal Gradient) *then, using an adaptation deterministically computed from* $(\mathrm{S}(x_k))$*, we have*

$$\|x_k - x^\star\|_2^2 = O_p\left(\left(1 - \lambda_{\min}(\mathrm{P}^\star)\frac{2\gamma\mu L}{\mu + L}\right)^k\right)$$

where $\mathrm{P}^\star$ *is the average projection matrix of the selection associated with* $\mathrm{S}(x^\star)$ *and* O_p *stands for Big O in probability,* i.e. *stochastic boundedness.*[35]

[35] *Alternatively, we know that there is a random time, finite almost surely, such that the rate is deterministically bounded by the quantity in parentheses after this time.*

Proof. Exact identification holds by Corollary 3.9 since (x_k) converges almost surely to $x^\star$, thus $\mathrm{S}(x_k)$ will exactly equal $\mathrm{S}(x^\star)$ in finite time. Now we go back to the proof of Theorem 5.16 to see that the random variable defined by

$$X_k = \begin{cases} x_{k_\ell} & \text{if } k \in (k_\ell, k_\ell + \mathrm{c}_\ell] \\ x_k & \text{if } k \in (k_\ell + \mathrm{c}_\ell, k_{\ell+1}] \end{cases} \quad \text{for some } \ell$$

also converges almost surely to $x^\star$. Intuitively, this sequence is a replica of (x_k) except that it stays fixed at the beginning of adaptation cycles when no adaptation is admitted. This means that $\mathrm{S}(X_k)$ which can be used for adapting the selection will exactly reach $\mathrm{S}(x^\star)$ in finite time. From that point on, since we use an adaptation technique that deterministically relies on $\mathrm{S}(x_k)$, there are no more adaptations and thus the rate matches the non-adaptive one of Theorem 5.6. Finally, using the almost sure finiteness of the identification time and Markov's inequality, we get the claimed result. □

This theorem means that if C and D are chosen in agreement with g, the adaptive algorithm ARPSD eventually reaches a linear rate in terms of iterations as the non-adaptive RPSD. In addition, the term $\lambda_{\min}(\mathrm{P})$ present in the rate now depends on the *final* selection and thus on the optimal structure which is much better than the structure-agnostic selection of RPSD in Theorem 5.6. In the next section, we develop practical rules for an efficient interlacing of g, C, and D.

5.3 Practical examples and discussion

Let us now see how Option 2 of Table 5.1 and the convergence results above can be used in practice.

5.3.1 Sparsity patterns & Coordinate-wise projections

We recall from Example 5.11, that for sparsity patterns, the structure can be captured by the collection

$$\mathrm{C} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\} \quad \text{with } \mathcal{M}_i = \{x \in \mathbb{R}^n : x^{[i]} = 0\},$$

and that a natural choice for the subspace is

$$\mathsf{D} = \{\mathcal{S}_1, \dots, \mathcal{S}_n\} \qquad \text{with } \mathcal{S}_i = \{x \in \mathbb{R}^n : x^{[j]} = 0 \ \ \forall j \neq i\}.$$

Then, a practical adaptive coordinate descent can be obtained from the following reasoning at each adaptation time $k = k_{\ell-1}$:

- Observe $\mathsf{S}(x_k)$ *i.e.* the support of x_k.
- Take all coordinates in the support and randomly select s coordinates outside the support.[36]. Compute the associated P_ℓ, Q_ℓ, and Q_ℓ^{-1}. Notice that $\lambda_{\min}(\mathsf{P}_\ell) = p_\ell = s/|\operatorname{null}(x_k)|$.
- Following the rules of Theorem 5.16, compute

$$\mathrm{c}_\ell = \left\lceil \frac{\log\left(\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\|_2^2\right) + \log(1/(1-\beta))}{\log(1/(1-\alpha_{\ell-1}))} \right\rceil \qquad \text{with } \alpha_{\ell-1} = 2p_{\ell-1}\gamma\mu L/(\mu+L)$$

  for some small fixed $0 < \beta \leq 2\gamma\mu L/(n(\mu+L)) \leq \inf_\ell \alpha_\ell$.
  Apply the new sampling after c_ℓ iterations (*i.e.* $k_\ell = k_{\ell-1} + \mathrm{c}_\ell$).

[36] We note by $\operatorname{null}(x)$ the set of indices of the null coordinates of x. For simplicity, we only consider the case $s \leq |\operatorname{null}(x_k)|$.

Example 5.18. Let us give a simple example in $\mathbb{R}^4$:

$$\text{for } x_k = \begin{pmatrix} 1.23 \\ -0.6 \\ 0 \\ 0 \end{pmatrix}, \mathsf{S}(x_k) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \text{ then } \begin{array}{l} \mathbb{P}[\mathcal{S}_1 \subseteq S_{k_\ell}] = \mathbb{P}[\mathcal{S}_2 \subseteq S_{k_\ell}] = 1 \\ \mathbb{P}[\mathcal{S}_3 \subseteq S_{k_\ell}] = \mathbb{P}[\mathcal{S}_4 \subseteq S_{k_\ell}] = p_\ell := s/|\operatorname{null}(x_k)| = s/2 \end{array}$$

$$\mathsf{P}_\ell = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & p_\ell & \\ & & & p_\ell \end{pmatrix} \quad \mathsf{Q}_\ell = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1/\sqrt{p_\ell} & \\ & & & 1/\sqrt{p_\ell} \end{pmatrix} \quad \mathsf{Q}_\ell^{-1} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{p_\ell} & \\ & & & \sqrt{p_\ell} \end{pmatrix}$$

◀

Finally, we notice that the above strategy with Option 2 of Table 5.1 produces moderate adaptations as long as the iterates are rather dense. To see this, observe first that $\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}$ is a diagonal matrix, the entries of which depend on the support of the corresponding coordinates at times $k_{\ell-1}$ and $k_{\ell-2}$. More precisely, the diagonal entries are described in the following table:

i is in the support at $k_{\ell-1}$	$k_{\ell-2}$	$\left[\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\right]_{ii}$
yes	yes	1
no	yes	$\frac{1}{p_\ell} = \frac{\lvert\operatorname{null}(x_{k_{\ell-1}})\rvert}{s}$
yes	no	$p_{\ell-1} = \frac{s}{\lvert\operatorname{null}(x_{k_{\ell-2}})\rvert}$
no	no	$\frac{p_{\ell-1}}{p_\ell} = \frac{\lvert\operatorname{null}(x_{k_{\ell-1}})\rvert}{\lvert\operatorname{null}(x_{k_{\ell-2}})\rvert}$

Thus, as long as the iterates are not sparse (*i.e.* in the first iterations, when $|\operatorname{null}(x_k)| \approx s$ is small), the adaptation cost is moderate so the first adaptations can be done rather frequently. Also, in the frequently-observed case when the support only decreases ($\mathsf{S}(x_{k_{\ell-2}}) \leq \mathsf{S}(x_{k_{\ell-1}})$), the second line of the table is not active and thus $\|\mathsf{Q}_\ell \mathsf{Q}_{\ell-1}^{-1}\| = 1$, so the adaptation can be done without waiting.

5.3.2 Vectors of fixed variations

We saw in Section 3.2.3 that for vectors of fixed variation, the structure can be displayed using that

$$\mathrm{C} = \{\mathcal{M}_1, .., \mathcal{M}_{n-1}\} \qquad \text{with } \mathcal{M}_i = \left\{x \in \mathbb{R}^n : x^{[i]} = x^{[i+1]}\right\},$$

then, the structure $\mathrm{S}(x_k)$ corresponds to the *jumps* of x_k (indeed $\mathrm{S}^{[i]}(x_k) = 0$ iff $x_k \in \mathcal{M}_i \Leftrightarrow x_k^{[i]} = x_k^{[i+1]}$).

In terms of sampling subspaces, a natural choice is

$$\mathrm{D} = \{\mathcal{S}_1, .., \mathcal{S}_{n-1}\} \qquad \text{with } \mathcal{S}_i = \left\{x \in \mathbb{R}^n : x^{[j]} = x^{[j+1]} \text{ for all } j \neq i\right\},$$

indeed, $\mathcal{M}_i \bigcap \mathcal{S}_i = \mathrm{span}(\{1\}) = \bigcap_j \mathcal{S}_j$ and $\mathcal{S}_i + \mathcal{M}_i = \mathbb{R}^n$.

The same reasoning as above can be done for vectors of fixed variation by using these families. At each adaptation time $k = k_{\ell-1}$:

- Observe $\mathrm{S}(x_k)$ *i.e.* the *jumps* of x;
- The adapted selection consists in selecting all jumps present in x_k and randomly selecting s jumps that are not in x_k. Compute P_ℓ, Q_ℓ, and Q_ℓ^{-1} (to the difference of coordinate sparsity they have to be computed numerically).
- For a fixed $\beta > 0$, compute

$$\mathrm{c}_\ell = \left\lceil \frac{\log\left(\|\mathrm{Q}_\ell \mathrm{Q}_{\ell-1}^{-1}\|_2^2\right) + \log(1/(1-\beta))}{\log(1/(1-\alpha_{\ell-1}))} \right\rceil.$$

  Apply the new sampling after c_ℓ iterations (*i.e.* $k_\ell = k_{\ell-1} + \mathrm{c}_\ell$).

5.4 Numerical illustrations

We report here some numerical illustrations for the behavior of our randomized proximal algorithms on standard problems involving ℓ_1 and 1D TV regularization.

5.4.1 Experimental setup

We consider the standard regularized logistic regression with three different regularization terms, which can be written for given $(\mathrm{a}_i, b_i) \in \mathbb{R}^{n+1}$ $(i = 1, \ldots, m)$ and parameters $\lambda_1, \lambda_2 > 0$

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + \exp\left(-b_i \mathrm{a}_i^\top x\right)\right) + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1 \|x\|_1 \tag{5.8a}$$

$$+ \lambda_1 \|x\|_{1,2} \tag{5.8b}$$

$$+ \lambda_1 \mathrm{TV}(x) \tag{5.8c}$$

We use two standard data-sets from the LibSVM repository: *a1a* ($m = 1,605$ $n = 123$) for the TV regularizer, and *rcv1_train* ($m = 20,242$ $n = 47,236$) for the ℓ_1 and $\ell_{1,2}$ regularizers. We fix the parameters $\lambda_2 = 1/m$ and λ_1 to reach a final sparsity of roughly 90%.

The subspace collections are taken naturally adapted to the regularizers: by coordinate for (5.8a) and (5.8b), and by variation for (5.8c). The adaptation strategies are the ones described in Section 5.3.

We consider five algorithms:[37]

Name	Reference	Description	Randomness
Proximal gradient	Chapter 4	vanilla proximal gradient descent	None
x RPCD	(Nesterov, 2012)	standard proximal coordinate descent	x coordinates selected for each update
x SEGA	(Hanzely et al., 2018)	Algorithm SEGA with coordinate sketches	$\mathrm{rank}(S_k)$ = x
x RPSD	Algorithm 5.1	(non-adaptive) random subspace descent	Option 2 of Table 5.1 with s = x
x ARPSD	Algorithm 5.2	adaptive random subspace descent	Option 2 of Table 5.1 with s = x

[37] In the following, x is often given in percentage of the possible subspaces, *i.e.* x% of $|\mathrm{D}|$, that is x% of n for coordinate projections and x% of $n-1$ for variation projections.

For the produced iterates, we measure the sparsity of a point x by $\|\mathrm{S}(x_k)\|_1$, which corresponds to the size of the supports for the ℓ_1 case and the number of jumps for the TV case. We also consider the quantity:

$$\text{Number of subspaces explored at time } k \;=\; \sum_{t=1}^{k} \|\mathrm{S}(x_t)\|_1.$$

We then compare the performance of the algorithms on three criteria:

- functional suboptimality vs iterations (standard comparison);
- size of the sparsity pattern vs iterations (showing the identification properties);
- functional suboptimality vs number of subspaces explored (showing the gain of adaptivity).

5.4.2 Illustrations for coordinate-structured problems

Comparison with standard methods

We consider first ℓ_1-regularized logistic regression (5.8a); in this setup, the non-adaptive RPSD boils down to the usual randomized proximal gradient descent. We compare the proximal gradient to its adaptive and non-adaptive randomized counterparts.

First, we observe that the iterates of Proximal gradient and ARPSD coincide. This is due to the fact that the sparsity of iterates only decreases ($\mathrm{S}(x_k) \leq \mathrm{S}(x_{k+1})$) along the convergence, and according to Option 2 all the non-zero coordinates are selected at each iteration and thus set to the same value as with Proximal gradient. However, a single iteration of 10%-ARPSD costs less in terms of number of subspaces explored, leading the speed-up of the right-most plot. Contrary to the adaptive ARPSD, the structure-blind RPSD identifies much later then Proximal gradient and shows poor convergence.

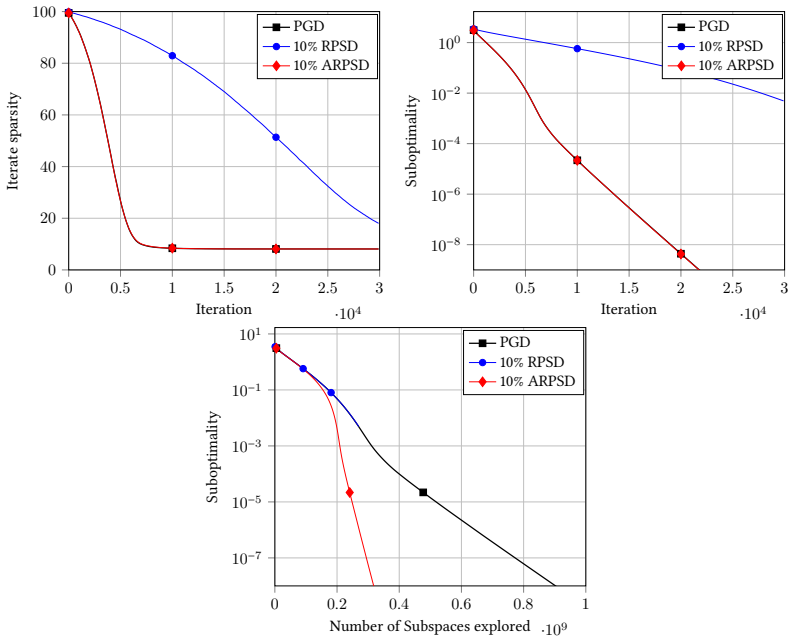


Figure 5.3: ℓ_1-regularized logistic regression (5.8a)

Comparison with SEGA

In Fig. 5.4, we compare ARPSD algorithm with SEGA algorithm featuring coordinate sketches (Hanzely et al., 2018). While the focus of SEGA is not to produce an efficient coordinate descent method but rather to use sketched gradients, SEGA and RPSD are similar algorithmically and reach similar rates. As mentioned in (Hanzely et al., 2018, Apx. G2), SEGA is slightly slower than plain randomized proximal coordinate descent (10% RPSD) but still competitive, which corresponds to our experiments. Thanks to the use of identification, ARPSD shows a clear improvement over other methods in terms of efficiency with respect to the number of subspaces explored.

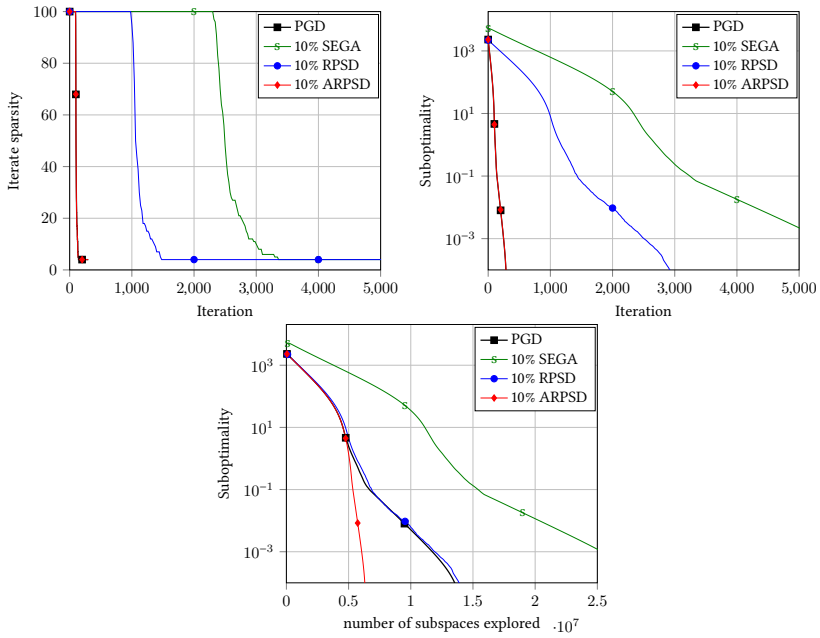


Figure 5.4: $\ell_{1,2}$ regularized logistic regression (5.8b)

5.4.3 Illustrations for total variation regularization

We focus here on the case of total variation (5.8c) which is a typical usecase for our adaptive algorithm and subspace descent in general. Fig. 5.5 displays a comparison between the vanilla proximal gradient and various versions of our subspace descent methods.

We observe first that RPSD, not exploiting the problem structure, fails to reach satisfying performances as it identifies lately and converges slowly. In contrast, the adaptive versions ARPSD perform similarly to the vanilla proximal gradient in terms of sparsification and suboptimality with respect to iterations. As a consequence, in terms of number of subspaces explored, ARPSD becomes much faster once a near-optimal structure is identified. More precisely, all adaptive algorithms (except 1 ARPSD, see the next paragraph) identify a subspace of size $\approx 8\%$ (10 jumps in the entries of the iterates) after having explored around 10^5 subspaces. Subsequently, each iteration involves a subspace of size 22, 32, 62 (out of a total dimension of 123) for 10%, 20%, 50% ARPSD respectively, resulting in the different slopes in the red plots on the rightmost figure.

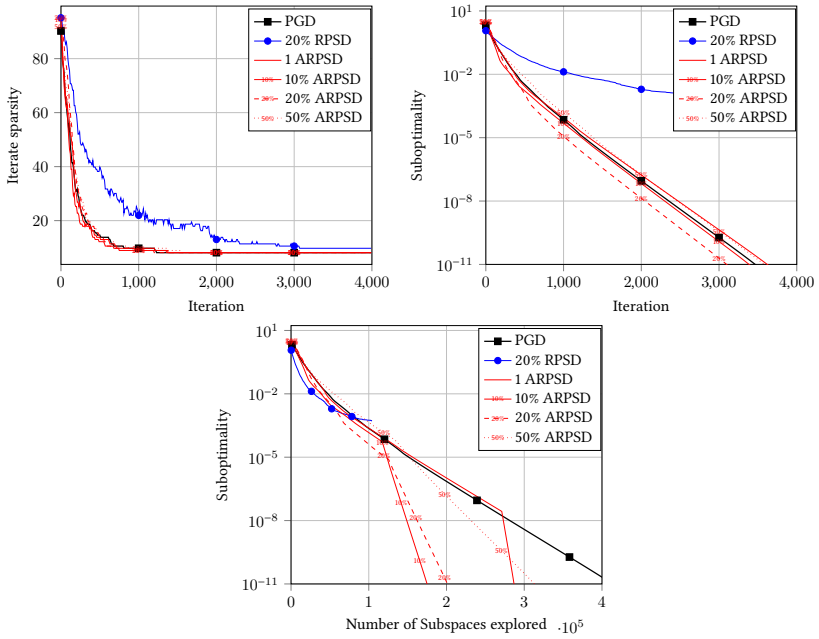


Figure 5.5: 1D-TV-regularized logistic regression (5.8c)

Finally, Fig. 5.6 displays 20 runs of 1 and 20% ARPSD as well as the median of the runs in bold. We notice that more than 50% of the time, a low-dimensional structure is quickly identified (after the third adaptation) resulting in a dramatic speed increase in terms of subspaces explored. However, this adaptation to the lower-dimensional subspace might take some more time (either because of poor identification in the first iterates or because a first heavy adaptation was made early and a pessimistic bound on the rate prevents a new adaptation in theory). Yet, one can notice that these adaptations are more stable for the 20% than for the 1 ARPSD, illustrating the "speed versus stability" tradeoff in the selection.

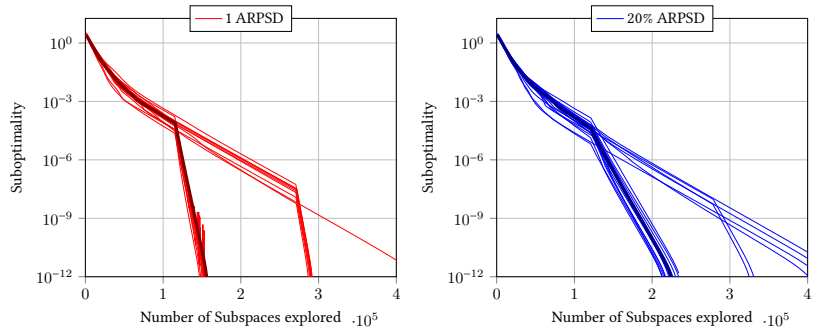


Figure 5.6: 20 runs of ARPSD and their median (in bold) on 1D-TV-regularized logistic regression (5.8c)

5.5 Concluding remarks

In this chapter, we saw how to adapt coordinate descent sampling strategies to the identified structure. The developed methods can efficiently reduce the "dimension" of the gradients updates (*e.g.* the number of gradient coordinates computed). They could be further improved by considering separately the Lipschitz constants per subspace or by considering the local conditioning of the problem once the optimal structure has been identified. The present results focus on the core idea to adapt sampling with structure.

We needed here only membership information from the structure. This notion is perfectly handled by *proximal qualification*, studied in Section 3.2, and does not need to rely on partial smoothness, developed in Sections 3.3 and 3.4. On the contrary, in the next chapter, we develop a method where local smoothness and stability of the identified manifold are directly used.

6 Riemannian Acceleration on Identified Manifolds

Time drowns in the unmeasured monotony of space.
Where uniformity reigns,
movement from point to point is no longer movement;
and where movement is no longer movement, there is no time.
Thomas Mann – *The Magic Mountain (1924)*

Leveraging the manifold identification properties of proximal gradient steps, we show that Riemannian Newton-like methods can be performed on the identified structure to drastically boost the convergence. For instance, we can prove the superlinear convergence of alternating proximal gradient and Riemannian steps when solving nondegenerated optimization problems. At the opposite of the previous chapter, we do not only use the identification properties but also the local smoothness once identification has happened.

This chapter is based on the following publication:

- Gilles Bareilles, Franck Iutzeler, Jérome Malick: Newton acceleration on manifolds identified by proximal-gradient methods, 2021.

Exploiting the underlying smooth substructure of objective functions to develop second-order methods has been a subject of fruitful and creative research in nonsmooth optimization, pioneered by the developments around $\mathcal{U}$-Newton algorithms (Lemaréchal et al., 2000) and the notion of partial smoothness (Lewis, 2002). Let us mention the $\mathcal{UV}$-Newton bundle method of (Mifflin and Sagastizábal, 2005) approximating the substructure from first-order information, and the recent k-bundle Newton method of (Lewis and Wylie, 2019) refining the approximation from a partial second-order oracle. Interestingly, these Newton-type methods for nonsmooth optimization are connected to the standard Newton methods of nonlinear programming (SQP) and to the Newton methods of Riemannian optimization; see (Miller and Malick, 2005). We refer to (Sagastizábal, 2018) for a recent review of these nonsmooth Newton methods relying on an implicit smooth substructure. In this chapter, we focus on a special situation where the smooth substructure can be detected and exploited numerically.

As in the previous chapters, we will consider problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x) \tag{$\mathcal{P}_{\mathrm{A}}$}$$

where f is a smooth differentiable function, and g is not everywhere differentiable but admits a *simple* proximal operator. More precisely, we assume that the proximal

operator of g outputs an explicit expression of the proximal point together with a representation of the current active manifold as previously considered in Section 3.5.

Since g has a simple proximal operator, first-order methods of choice to minimize F are variants of the proximal gradient method as investigated in Chapter 4. We saw that these algorithms were able to identify the nonsmooth substructure of F in finite time (see Section 4.2.1 and in particular Theorem 4.10) but, unfortunately, the user never knows if the current manifold is the optimal one or not. Adaptivity is thus once again key in the exploitation of proximally discovered substructure.

We propose here a Newton acceleration of the proximal-gradient algorithm solving the nonsmooth optimization problem ($\mathcal{P}_A$). Our algorithm uses the same basic ingredients that work behind the scenes for the existing nonsmooth Newton algorithms recalled above (*e.g.* (Daniilidis et al., 2006; Lewis and Wylie, 2019; Mifflin and Sagastizábal, 2005)). In our context, they become apparent, and we can then heavily rely on recent developments of Riemannian optimization (Absil et al., 2009; Boumal, 2020). Specifically our algorithm relies on (i) explicit proximal operations for structure identification and (ii) the efficiency of Riemannian Newton-type methods to finally benefit from faster convergence. We present a convergence analysis showing superlinear convergence of the resulting algorithm under some qualification assumptions – but without prior knowledge on the final manifold.

6.1 A general algorithm for explicit nonsmooth problems

Let us start by specifying our blanket assumptions for this chapter on the composite problem ($\mathcal{P}_A$).

Assumption 6.1 (On the functions)**.** The functions f and g are proper and

i) $f : \mathbb{R}^n \to \mathbb{R}$ is C^2 with an L-Lipschitz continuous gradient;

ii) $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is lower semi-continuous;

iii) $\operatorname{prox}_{\gamma g}$ is non-empty on $\mathbb{R}^n$ for any $\gamma > 0$;

iv) $F := f + g$ is bounded below.

These assumptions are mostly common except the third point which directly comes from our idea of using the proximal operator both for the optimization itself and as an oracle for the current structure of the iterates.[38]

[38] See Section 2.2 for more details about the existence of the proximity operator.

6.1.1 Proximal gradient with Riemannian acceleration

Building on the identification properties of proximal methods (see Chapter 3 and Section 4.2.1), we leverage this ability to an algorithmic advantage by reducing our working space to the identified structure. "Smooth" structures (involving smooth manifolds and smooth restrictions on it) are of special interest and open the way to Newton acceleration.

Algorithm 6.1 Proximal gradient with Riemannian acceleration

1: **repeat**
2: Compute $x_k \in \operatorname{prox}_{\gamma g}(y_{k-1} - \gamma \nabla f(y_{k-1}))$ and get $\mathcal{M}_k \ni x_k$
3: Update $y_k = \operatorname{ManUp}_{\mathcal{M}_k}(x_k)$ on the current manifold
4: **until** stopping criterion

We propose the general algorithm (Algorithm 6.1) which consists in, first, performing a proximal gradient step $x_k \in \operatorname{prox}_{\gamma g}(y_{k-1} - \gamma \nabla f(y_{k-1}))$ that provides both the current point x_k and the manifold $\mathcal{M}_k$ where it lies,[39] and, second, carrying out a Riemannian optimization update, $\mathrm{ManUp}_{\mathcal{M}_k}$, on the current manifold. This algorithm is general in the sense that we do not precise for now what is the Riemannian step ManUp. Our plan of action is to lay out the generic assumptions on the manifold updates that provide global and local convergence, respectively in Section 6.1.2 and 6.1.3. Then, we will investigate in Section 6.2 the type of Riemannian Newton methods that fall into this scheme.

[39] This implicitly assumes that the computation of the proximity operator comes with the knowledge of the structure of the output, which is the case for many cases of interest in this part; see Sections 3.2.3 and 3.5.

Remark 6.2 (Direct extensions of Algorithm 6.1). The first step (the proximal gradient update) is solely responsible for the identification of the current manifold. Thus, strategies other than alternating proximal gradient and Riemannian steps could be considered as long as infinitely many proximal gradient steps are performed; the results layed out next can be easily adapted to such situations.

The proximal gradient step could also be replaced by another optimization method, as long as it identifies and converges; however, the changes in the forthcoming results would be more significant. ◀

6.1.2 Global convergence

The following result show that Algorithm 6.1 converges to a critical value of F and that all accumulation points of its iterates are critical points. In order for this to hold, we only need the mild assumption that the manifold update does not increase the functional value (this offers a broad choice of methods since this kind of descent is easily obtained by line-search as discussed in Section 6.2.1).

Theorem 6.3 (Global convergence). *Let Assumption 6.1 hold and take $\gamma \in (0, \frac{1}{L})$. Suppose that the manifold update $\mathrm{ManUp}_{\mathcal{M}}$ provides descent, that is for any x in $\mathcal{M}$*

$$F(\mathrm{ManUp}_{\mathcal{M}}(x)) \leq F(x).$$

Then, Algorithm 6.1 generates non-increasing functional values ($F(x_{k+1}) \leq F(y_k) \leq F(x_k)$ for all k) and all limit points of (x_k) and (y_k) are critical points of F that share the same functional value.

Proof. The proximal gradient steps provides a descent (see Section 2.3.3) thus choosing y_k such that $F(y_k) \leq F(x_k)$ (by assumption on the manifold update) yields:

$$F(x_{k+1}) \overset{\text{Lemma 2.39}}{\leq} F(y_k) - \frac{1-\gamma L}{2\gamma}\|x_{k+1} - y_k\|^2 \leq F(x_k) - \frac{1-\gamma L}{2\gamma}\|x_{k+1} - y_k\|^2. \quad (6.1)$$

The sequence $(F(x_k))$ is thus non-increasing and lower-bounded, therefore it converges. Besides, any accumulation point of (x_k) is a critical point of F. Indeed, summing equation (6.1) for $k = 1, \ldots, n$ yields:

$$\frac{1-\gamma L}{2\gamma} \sum_{k=0}^{n} \|x_{k+1} - y_k\|^2 \leq F(x_0) - F(x_{n+1}) \leq F(x_0) - \inf F < +\infty.$$

Therefore the general term of the above series $\|x_{k+1} - y_k\|^2$ converges to 0, which implies, by Lemma 2.41, that the distance from $\partial F(x_k)$ to 0 converges to 0. The outer-semi continuity property of the limiting subdifferential allows to conclude that every

accumulation point of (x_k) is a critical point of F. Finally, all limit points share the same functional value as F is lower semi-continuous. □

We now focus on our main case of interest where (i) the objective is partly smooth at some critical point with respect to a manifold on which (ii) the Riemannian update is locally super-linearly convergent. The first point is natural when the nonsmooth objective F admits a smooth substructure, which often appears in regularized regression problems. The second point means relying on a higher-order Riemannian method to fully exploit the smoothness of the identified substructure, which is our primary motivation.

6.1.3 *Identification, local smoothness, and super-linear convergence*

Naturally, we build upon the analysis on partial smoothness carried in Sections 3.3 and 3.4. More particularly, in order to place ourselves in the context of Theorem 3.18, which captures the localization properties of the proximal gradient operator, we introduce the notion of *r-structured critical points*.

Definition 6.4. A point $\bar{x}$ of a C^2 manifold $\mathcal{M}$ is a *r-structured critical* point for (f, g) if:

i) proximal gradient stability: $\bar{x} = \operatorname{prox}_{g/r}(\bar{x} - 1/r\nabla f(\bar{x}))$;

ii) qualification condition: $0 \in \operatorname{ri}(\nabla f + \partial g)(\bar{x})$;

iii) prox-regularity: g is r-prox-regular at $\bar{x}$;

iv) partial smoothness: g is partly-smooth at $\bar{x}$ with respect to $\mathcal{M}$.

Definition 6.4 gives a precise characterization of the conditions allowing a critical point to benefit from a local structure stability by the proximal gradient operator. An illustration of this structure is shown in Fig. 6.1. While points ii,iii,iv are rather standard in the literature (see *e.g.* (Daniilidis et al., 2006)), point i is less standard. First, this point is directly verified when g is convex, in which case any positive r is valid. In the nonconvex case, this condition is not often laid out precisely (still, it appears in the notion of identifiability in (Drusvyatskiy and Lewis, 2014)) but it is necessary to take a stepsize range of $(0, 1/r)$ in the proximal gradient around $\bar{x}$.[40]

[40] If point i is verified for some stepsize $\bar{\gamma}$, *i.e.* $\bar{x} = \operatorname{prox}_{\bar{\gamma}g}(\bar{x} - \bar{\gamma}\nabla f(\bar{x}))$, then the maximal stepsize is reduced to $\min(\bar{\gamma}, 1/r)$.

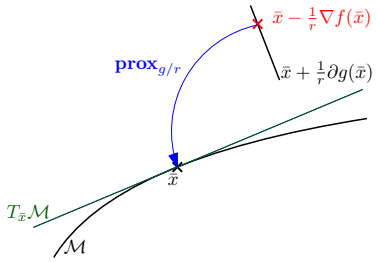


Figure 6.1: Illustration of a r-structured critical point. Point i is illustrated by the blue arrow, while point ii implies that the red cross is in the interior of the black segment. Partial smoothness appears in the fact that the black segment is perpendicular to the tangent plane of $\mathcal{M}$ at $\bar{x}$ (see also Fig. 3.3).

With this definition, Theorem 3.18 gives the following corollary. Note that the manifold is defined unambiguously since it is locally unique by partial smoothness, see Proposition 3.14.

Corollary 6.5. *Let f be a C^2 function on $\mathbb{R}^n$ and g a lower semi-continuous function on $\mathbb{R}^n$. Take $\bar{x} \in \mathcal{M}$ a r-structured critical point for (f, g). Then, for any $\gamma \in (0, \frac{1}{r})$, the proximal gradient map $y \mapsto \mathrm{prox}_{\gamma g}(y - \gamma \nabla f(y))$ is C^1 and $\mathcal{M}$-valued near $\bar{x}$.*
In particular, if a sequence (y_k) satisfies $y_k \to \bar{x}$, then $x_k := \mathrm{prox}_{\gamma g}(y_k - \gamma \nabla f(y_k)) \in \mathcal{M}$ for k large enough.

Using the structure identification results above, we can guarantee that our method benefits from superlinear convergence, provided that the considered Riemannian method is superlinearly convergent locally around a limit point.

Theorem 6.6. *Let Assumption 6.1 hold and take $\gamma \in (0, \frac{1}{L})$. Assume that Algorithm 6.1 generates a sequence (y_k) which admits at least one limit point $\bar{x}$ such that:*

i) $\bar{x} \in \mathcal{M}$ is a r-structured critical point for (f, g) with $r < \frac{1}{\gamma}$;

ii) $\mathrm{ManUp}_{\mathcal{M}}$ has superlinear convergence rate of order $1 + \theta \in (1, 2)$ on a neighborhood of $\bar{x}$ in $\mathcal{M}$.

Then, after some finite time:

a) the sequence (x_k) lies on $\mathcal{M}$;

b) (x_k) converges to $\bar{x}$ superlinearly with the same order as ManUp:

$$\mathrm{dist}_{\mathcal{M}}(x_{k+1}, \bar{x}) \leq c\, \mathrm{dist}_{\mathcal{M}}(x_k, \bar{x})^{1+\theta} \qquad \text{for some } c > 0.$$

Proof. Let us note $\mathsf{T}(y) = \mathrm{prox}_{\gamma g}(y - \gamma \nabla f(y))$ for $y \in \mathbb{R}^n$ (in all generality, this would be a set-valued mapping but we will only call it at points for which it is single-valued).

The part i of the assumptions enables us to show the existence of some neighborhood of $\bar{x}$ on which the proximal gradient operation is $\mathcal{M}$-valued and Lipschitz continuous. More precisely, Corollary 6.5 tells us that there exists $\delta_1 > 0$ and $C > 0$ such that, for any y in $\mathcal{B}(\bar{x}, \delta_1)$, we have

$$\mathsf{T}(y) \in \mathcal{M} \qquad \text{and} \qquad \|\mathsf{T}(y) - \mathsf{T}(\bar{x})\| \leq C\|y - \bar{x}\|.$$

Now, if y additionally belongs to $\mathcal{M}$, we immediately deduce that there exists $\varepsilon_1 > 0$ such that for any y in $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon_1)$, $\mathsf{T}(y) \in \mathcal{M}$;[41] but in addition, the Euclidean Lipchitz continuity can be translated into a Riemannian one (see Lemma 2.47) since for some $\delta > 0$,

$$\begin{aligned}(1-\delta)\mathrm{dist}_{\mathcal{M}}(\mathsf{T}(y), \bar{x}) = (1-\delta)\mathrm{dist}_{\mathcal{M}}(\mathsf{T}(y), \mathsf{T}(\bar{x})) &\leq \|\mathsf{T}(y) - \mathsf{T}(\bar{x})\| \\ &\leq C\|y - \bar{x}\| \leq C(1+\delta)\mathrm{dist}_{\mathcal{M}}(y, \bar{x}) \end{aligned} \tag{6.2}$$

Hence, there is $q_1 > 0$ such that for any y in $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon_1)$

$$\mathrm{dist}_{\mathcal{M}}(\mathsf{T}(y), \bar{x}) = \mathrm{dist}_{\mathcal{M}}(\mathsf{T}(y), \mathsf{T}(\bar{x})) \leq q_1\, \mathrm{dist}_{\mathcal{M}}(y, \bar{x}). \tag{6.3}$$

Then, the part ii of the assumptions gives us the existence of ε_2, $q_2 > 0$ and $\theta \in (0, 1)$ such that, for any x in $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon_2)$,

$$\mathrm{dist}_{\mathcal{M}}(\mathrm{ManUp}_{\mathcal{M}}(x), \bar{x}) \leq q_2\, \mathrm{dist}_{\mathcal{M}}(x, \bar{x})^{1+\theta}. \tag{6.4}$$

Let us now take any $x \in \mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon)$ where $\varepsilon = \min(\varepsilon_1, \varepsilon_2, (\varepsilon_1/q_2)^{\frac{1}{1+\theta}}, (q_2 q_1)^{-\frac{1}{\theta}})$:
(1) Since $x \in \mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon_2)$, the manifold update (6.4) yields

$$\mathrm{dist}_{\mathcal{M}}(\mathrm{ManUp}_{\mathcal{M}}(x), \bar{x}) \leq q_2\, \mathrm{dist}_{\mathcal{M}}(x, \bar{x})^{1+\theta} \leq q_2\, \varepsilon^{1+\theta} \leq \varepsilon_1.$$

[41] We note $\mathcal{B}_{\mathcal{M}}(x, \varepsilon) = \{u \in \mathcal{M} : \mathrm{dist}_{\mathcal{M}}(u, x) \leq \varepsilon\}$ the points in $\mathcal{M}$ at a (Riemannian) distance to x smaller than ε.

(2) As $\mathrm{ManUp}_{\mathcal{M}}(x)$ lies in $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon_1)$, the proximal gradient update (6.3) applied to $y = \mathrm{ManUp}_{\mathcal{M}}(x)$ gives

$$\begin{aligned} \mathrm{dist}_{\mathcal{M}}(\mathsf{T}(\mathrm{ManUp}_{\mathcal{M}}(x)), \bar{x}) &\leq q_1 \mathrm{dist}_{\mathcal{M}}(\mathrm{ManUp}_{\mathcal{M}}(x), \bar{x}) \\ &\leq q_1 q_2 \ \mathrm{dist}_{\mathcal{M}}(x, \bar{x})^{1+\theta} \\ &\leq q_1 q_2 \ \varepsilon^{\theta} \mathrm{dist}_{\mathcal{M}}(x, \bar{x}). \end{aligned} \tag{6.5}$$

Since $q_2 q_1 \varepsilon^{\theta} \leq 1$ by construction, this allows us to conclude that for any $x \in \mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon)$, we have

$$\mathrm{dist}_{\mathcal{M}}(\mathsf{T}(\mathrm{ManUp}_{\mathcal{M}}(x)), \bar{x}) \leq \mathrm{dist}_{\mathcal{M}}(x, \bar{x}). \tag{6.6}$$

We have thus proved the existence of a neighborhood $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon)$ of $\bar{x}$ in $\mathcal{M}$ which is stable for an iteration of Algorithm 6.1 and over which one iteration has a superlinear improvement of order $1 + \theta$ (by (6.5)).

Finally, since $\bar{x}$ is a limit point of (y_k):

- an iterate will reach any (Euclidean) ball around $\bar{x}$ in finite time, notably there exists $K < \infty$ such that $y_K \in \mathcal{B}(\bar{x}, (1 - \delta)\varepsilon/C)$;
- equation (6.2) then tells us that $\mathrm{dist}_{\mathcal{M}}(\mathsf{T}(y_K), \bar{x}) \leq \varepsilon$ and thus x_k and y_k belong to $\mathcal{B}_{\mathcal{M}}(\bar{x}, \varepsilon)$ for all $k > K$ by (6.6) and the definition of ε.

We may thus conclude that $x_{k+1} = \mathsf{T}(y_k) \in \mathcal{M}$ for all $k \geq K$, and, using (6.5),

$$\mathrm{dist}_{\mathcal{M}}(x_{k+1}, \bar{x}) \leq q \ \mathrm{dist}_{\mathcal{M}}(x_k, \bar{x})^{1+\theta}$$

for all $k > K$ and $q = q_1 q_2$. □

6.2 Newton acceleration

In this section, we investigate several possibilities for the manifold updates in Algorithm 6.1. Since the procedure $\mathrm{ManUp}_{\mathcal{M}}$ is required not to degrade function value (for global convergence, see Theorem 6.3), we first study in Section 6.2.1 the use of line-search in our context. Then, we show in Sections 6.2.2 and 6.2.3 how to use Riemannian (truncated) Newton method within our framework and derive superlinear/quadratic convergence guarantees to qualified critical points.

6.2.1 Ensuring functional descent while preserving local rates: Linesearches

As developed in the previous section, the Riemannian update $\mathrm{ManUp}_{\mathcal{M}}$ should produce an update that (i) does not degrade the function value and (ii) enjoys a superlinear local convergence rate. We thus consider a simple (Riemannian) line-search and we prove that, under mild assumptions, it can find a point which decreases the function value while retaining the superior local properties. Surprisingly, this result does not appear in the standard references on Riemannian optimization. We provide here the necessary developments inspired from the classical textbook (Dennis Jr and Schnabel, 1996).

Standing at point $x \in \mathcal{M}$ with a proposed direction $\eta \in T_x\mathcal{M}$, a stepsize $\alpha > 0$ is *acceptable* is it satisfies the following *Armijo* condition:

$$F(\mathrm{R}_x(\alpha\eta)) \leq F(x) + m_1 \alpha \langle \mathrm{grad}\, F(x), \eta \rangle, \qquad \text{for some } 0 < m_1 < 1/2. \quad \text{(Amijo rule)}$$

When η is a descent direction, *i.e.* when $\langle \mathrm{grad}\, F(x), \eta \rangle < 0$, this line search ensures functional decrease. The condition for existence of stepsizes α satisfying the Armijo rule can be obtained by following (Dennis Jr and Schnabel, 1996, Sec 6.3).

Lemma 6.7. *Let Assumption 6.1 hold and consider a manifold $\mathcal{M}$ equipped with a retraction R and a pair $(x, \eta) \in T\mathcal{B}$. If F is differentiable on $\mathcal{M}$ at x and $\langle \operatorname{grad} F(x), \eta \rangle < 0$, then there exists $\hat{\alpha} > 0$ such that any step size $\alpha \in (0, \hat{\alpha})$ is acceptable by the Amijo rule.*

Proof. We adapt a part of the proof of (Dennis Jr and Schnabel, 1996, Th. 6.3.2) for the Armijo rule and the Riemannian setting. Since $m_1 < 1/2$, for any α sufficiently small there holds

$$F \circ \mathrm{R}_x(\alpha\eta) \leq F \circ \mathrm{R}_x(0) + m_1 \, \mathrm{D}\,(F \circ \mathrm{R}_x)\,(0)[\alpha\eta] = F(x) + m_1 \alpha \langle \operatorname{grad} F(x), \eta \rangle.$$

Since F is bounded below, there exists some smallest positive $\hat{\alpha}$ such that $F(\mathrm{R}_x(\hat{\alpha}\eta)) = F(x) + m_1 \hat{\alpha} \langle \operatorname{grad} F(x), \eta \rangle$. Thus all stepsizes in $(0, \hat{\alpha})$ are acceptable by the Amijo rule. □

In addition, near a critical point and with a Newton direction (or in general a superlinearly converging method), the linesearch should accept the unit stepsize so that a full step may be taken. This is the case when the Riemannian Hessian is positive definite at this point.

Lemma 6.8. *Let Assumption 6.1 hold and consider a manifold $\mathcal{M}$, a point $x^\star \in \mathcal{M}$, and a pair $(x, \eta) \in T\mathcal{B}$. Assume that F is twice differentiable on $\mathcal{M}$ near $x^\star$ and that $x^\star$ is a local minimizer on $\mathcal{M}$ with $\operatorname{Hess} F(x^\star)$ is positive definite.*
If the direction η brings a superlinear improvement towards $x^\star$, that is if $\operatorname{dist}_{\mathcal{M}}(\mathrm{R}_x(\eta), x^\star) = o(\operatorname{dist}_{\mathcal{M}}(x, x^\star))$ as $x \to x^\star$, then η is acceptable by the Amijo rule with unit stepsize $\alpha = 1$.

Proof. The proof of (Bonnans et al., 2006, Th. 4.16) can be adapted to the Riemannian setting. The full proof can be found in (Bareilles et al., 2020a, Apx. B.2). □

We thus consider a *backtracking* linesearch for finding an acceptable stepsize α. The unit stepsize is first tried, and then the search space is reduced geometrically so that superlinear steps are taken whenever possible. The procedure terminates within a finite number of iterations. Besides, the backtracking nature of the linesearch ensures that step sizes are not overly small. In practice, we use exactly (Dennis Jr and Schnabel, 1996, Alg. A6.3.1), which features polynomial interpolation of F in the search space.

Remark 6.9 (Other types of linesearch). We do not consider a more sophisticated linesearch, such as Armijo-Wolfe, as it requires the gradient of $F \circ \mathrm{R}_x$ at $y \neq x$, which requires to differentiate the retraction. This operation is not simple, we prefer to avoid it, as in (Huang et al., 2018).

Besides, the additional Wolfe condition serves in part to ensure that not-too-small steps are taken, which is not ensured theoretically by the Armijo rule but the backtracking nature of the linesearch mitigates this effect in practice. ◀

6.2.2 Riemannian Newton & quadratic convergence

We now construct a manifold update based on the Riemannian Newton method (Absil et al., 2009, Chap. 6), this is the simplest method enjoying a local quadratic convergence rate. It consists in finding a direction $d \in T_x\mathcal{M}$ that minimizes the second order model

Algorithm 6.2 ManUp-Newton

Require: Manifold $\mathcal{M}$, point $x \in \mathcal{M}$.
1: Find d in $T_x\mathcal{M}$ that solves the Newton equation
2: Find α satisfying the Amijo rule with direction d by backtracking.
3: **Return:** $y = \mathrm{R}_x(\alpha d)$

(2.18) of F at point $x \in \mathcal{M}$, or equivalently that solves Newton equation (see (Boumal, 2020, Sec. 6.2)):

$$\operatorname{grad} F(x) + \operatorname{Hess} F(x)[d] = 0. \qquad \text{(Newton equation)}$$

In order to ensure that the Newton direction is well-defined and provides descent, we assume that the Riemannian Hessian of F is positive definite at each iterate (relative to the working manifold).[42]

[42] this is verified for instance if F is strongly convex in the whole space.

Theorem 6.10. *Let Assumption 6.1 hold and take $\gamma \in (0, \frac{1}{L})$. Consider the sequence of iterates (x_k) generated by Algorithm 6.1 equipped with the Riemannian Newton manifold update (Algorithm 6.2). If $\operatorname{Hess} F(x_k)$ is positive definite at each step, then all limit points of (x_k) are critical points of F and share the same functional value.*
Furthermore, assume that the sequence (y_k) admits at least one limit point $x^\star$ such that

i) $x^\star \in \mathcal{M}$ is a r-structured critical point for (f, g) with $r < \frac{1}{\gamma}$;

ii) $\operatorname{Hess}_{\mathcal{M}} F(x^\star) \succ 0$ and $\operatorname{Hess}_{\mathcal{M}} F$ is locally Lipschitz around $x^\star$.

Then, after some finite time,

a) the sequence (x_k) lies on $\mathcal{M}$;

b) x_k converges to $x^\star$ quadratically: there exist $c > 0$,

$$\operatorname{dist}_{\mathcal{M}}(x_{k+1}, x^\star) \leq c \operatorname{dist}_{\mathcal{M}}(x_k, x^\star)^2.$$

Proof. As the Riemannian hessian is assumed to be positive definite, Newton's direction is a descent direction:

$$\langle \operatorname{grad} F(x_k), d_k \rangle = -\langle \operatorname{grad} F(x_k), \operatorname{Hess} F(x_k)^{-1} \operatorname{grad} F(x_k) \rangle < 0.$$

The Riemannian Newton manifold step is therefore well-defined, and stepsizes acceptable by the linesearch exist by Lemma 6.7, so that the manifold update is well defined and provides descent. Thus, Theorem 6.3 ensures that every accumulation point of the iterate sequence is a critical point for F.

Furthermore, assumption ii ensures that the Riemannian Newton direction d computed in step 1 of Algorithm 6.2 provides a quadratic improvement towards $x^\star$ on a neighborhood of $x^\star$ on $\mathcal{M}$. Indeed, (Absil et al., 2009, Th. 6.3.2) provides the existence of a neighborhood over which

$$\operatorname{dist}_{\mathcal{M}}(\mathrm{R}_x(d), x^\star) \leq (\beta\gamma_J + \gamma_R)\operatorname{dist}_{\mathcal{M}}(x, x^\star)^2 + O(\operatorname{dist}_{\mathcal{M}}(x, x^\star)^3),$$

where γ_J and γ_R quantify smoothness of Hess and R_x as defined in the proof.[43]

[43] We specialize the result to normal coordinates, that is $\varphi = \exp_{x^\star}$, which allows to write estimates in terms of Riemannian distances and make the Christoffel terms becomes negligible, see (Absil et al., 2009, p. 116).

Finally, the linesearch returns the unit-stepsize after some finite time: $\alpha = 1$ is tried first, and is acceptable for direction providing superlinear improvement by Lemma 6.8. Thus the whole Riemannian Newton update provides quadratic improvement after some finite time. Using this and assumption i, Theorem 6.6 applies and yields the results. □

This theorem states that alternating proximal gradient steps and Riemannian Newton steps on the identified manifold converges quadratically to structured points with virtually the same assumptions as the Euclidean Newton method. Notably, a point $x^\star \in \mathcal{M}$ that is both r-structured critical for (f, g) and such that $\operatorname{Hess}_{\mathcal{M}} F(x^\star) \succ 0$ is a strong local minimizer of F on $\mathbb{R}^n$, that is there exists some $\varepsilon > 0$ such that, for x near $x^\star$,

$$F(x) \geq F(x^\star) + \frac{\varepsilon}{2}\|x - x^\star\|^2.$$

However, we can notice two issues of Newton's method (present in both the Euclidean and Riemannian settings): (i) at each iteration a linear system has to be solved to produce the Newton direction d; and (ii) the direction d does not always provide descent without the strong assumption that the Riemannian Hessian is positive definite at each step. Truncated versions of Newton's method overcome these issues, as we see for our context, in the next section.

6.2.3 Riemannian Truncated Newton & superlinear convergence

(Riemannian) Truncated Newton consists in solving the Newton equation partially by using a (Riemannian) conjugate gradient procedure so that whenever the resolution is stopped, the resulting direction provides descent on the function. The successive iterates of the conjugate gradient thus slide from the negative gradient towards the exact solution of Newton's equation as the number of iterations grow, while maintaining the descent property. This method was proposed first by Dembo and Steinhaug (Dembo and Steihaug, 1983) as a specific case of inexact Newton method (Dembo et al., 1982), (Nocedal and Wright, 2006) discusses it as *line search Newton-CG*, and a review can be found in (Knoll and Keyes, 2004).

The quality of the truncated Newton direction is controlled by a parameter $\eta \in [0, 1)$ which bounds the ratio of residual and gradient norms:

$$\| \operatorname{grad} F(x) + \operatorname{Hess} F(x)[d] \| \leq \eta \| \operatorname{grad} F(x) \|. \qquad \text{(Inexact Newton equation)}$$

Taking $\eta = 0$ allows only the exact Newton step, while $\eta = 1$ allows a broad set of directions, including $d = 0$.

Algorithm 6.3 ManUp-Newton-CG

Require: Manifold $\mathcal{M}$, point $x \in \mathcal{M}$, convergence defining parameter $\theta \in (0, 1]$.

1: Let $\eta = \| \operatorname{grad} F(x) \|^\theta$
2: Find d that solves the Inexact Newton equation
3: Find α satisfying the Amijo rule with direction d by backtracking.
4: **Return:** $y = \mathrm{R}_x(\alpha d)$

Theorem 6.11. *Let Assumption 6.1 hold and take $\gamma \in (0, \frac{1}{L})$. Consider the sequence of iterates (x_k) generated by Algorithm 6.1 equipped with the Riemannian Truncated Newton manifold update (Algorithm 6.3). Then all limit points of (x_k) are critical points of F and share the same function value.*
Furthermore, assume that sequence (y_k) admits at least one limit point $x^\star$ such that

i) $x^\star \in \mathcal{M}$ is a r-structured critical point for (f, g) with $r < \frac{1}{\gamma}$;

ii) $\operatorname{Hess}_{\mathcal{M}} F(x^\star) \succ 0$ and $\operatorname{Hess}_{\mathcal{M}} F$ is locally Lipschitz around $x^\star$.

iii) $\eta_k = O(\|\operatorname{grad} F(x_k)\|^\theta)$, *for some* $\theta \in (0, 1]$.

Then,

a) for k *large enough, the full sequence* (x_k) *lies on* $\mathcal{M}$*;*

b) x_k *converges to* $x^\star$ *superlinearly with order* $1 + \theta$*: for large* k*, there exist* $c > 0$*,*

$$\operatorname{dist}_{\mathcal{M}}(x_{k+1}, x^\star) \leq c \operatorname{dist}_{\mathcal{M}}(x_k, x^\star)^{1+\theta}.$$

Proof. The first step is to show that the direction provided by the approximate resolution of the Inexact Newton equation is a descent direction. Following the analysis of (Dembo and Steihaug, 1983, Lemma A.2) on the euclidean space $T_x\mathcal{M}$, we obtain that if F is twice differentiable on $\mathcal{M}$ at x and if x is not a stationary point of F, then

$$\langle \operatorname{grad} F(x), d\rangle \leq -\min(1, \|\operatorname{Hess} F(x)\|^{-1})\|\operatorname{grad} F(x)\|^2,$$

where d was obtained solving the Inexact Newton equation with any forcing parameter η.

Thus, stepsizes acceptable by the linesearch exist by Lemma 6.7, so that the Riemannian Newton-CG manifold update is well defined and provides descent. Thus, Theorem 6.3 ensures that every accumulation point of the iterate sequence is a critical point for F.

Furthermore, assumptions ii and iii ensure that the Riemannian Newton CG direction d computed in step 1 of Algorithm 6.3 provides a superlinear improvement towards $x^\star$ on a neighborhood of $x^\star$ on $\mathcal{M}$. Indeed, (Absil et al., 2009, Th. 8.2.1), provides the existence of a neighborhood over which

$$\operatorname{dist}_{\mathcal{M}}(\operatorname{ManUp}(x), x^\star) \leq 2\beta\gamma_\xi^{1+\theta}\operatorname{dist}_{\mathcal{M}}(x, x^\star)^{1+\theta} + C\operatorname{dist}_{\mathcal{M}}(x, x^\star)^2,$$

where γ_ξ is a Lipschitz constant for $\operatorname{grad} F$ near $x^\star$ and C, β denote two positive constants.[44]

This means that the linesearch returns a unit stepsize after some finite time ($\alpha = 1$ is tried first, and is acceptable for direction providing superlinear improvement by Lemma 6.8). Thus, the whole Riemannian Newton-CG update eventually provides fast improvement. Using this and assumption i, Theorem 6.6 applies and yields the results. □

[44] Again, we have specialized the result to normal coordinates, that is $\varphi = \exp_{x^\star}$, which allows to write estimates in terms of Riemannian distances.

6.3 Numerical illustrations

In this section, we illustrate the effect of Newton acceleration, in terms of identification of the final manifold and local convergence.[45]

We consider Algorithm 6.1 equipped with either the Newton update of Algorithm 6.2, denoted 'Alt. Newton' or the truncated Newton update of Algorithm 6.3, denoted 'Alt. Truncated Newton'. Both algorithms use a Conjugate Gradient procedure to solve the Newton equation, either exactly or not. Each CG iteration requires one (Riemannian) Hessian-vector product, avoiding to form a Hessian matrix. These methods are compared to the Proximal Gradient and the Accelerated Proximal Gradient, which serve as baseline.

We report the numerical results in figures showing a) the suboptimality $F(x_k) - F(x^\star)$ of the current iterate x_k versus time, and b) the dimension of the current manifold $\mathcal{M}_k \ni x_k$ versus iteration. We also report a table comparing the algorithms at the first

[45] The algorithms and problems are implemented in Julia (Bezanson et al., 2017); experiments may be reproduced using the code available at https://github.com/GillesBareilles/NewtonRiemannAccel-ProxGrad

iteration that makes suboptimality lower than tolerances 10^{-3} and 10^{-9} for various measures summarized in the following table:

$F(x_k) - F(x^\star)$	Suboptimality at current iteration.
#prox. grad. steps	Number of proximal gradient steps, each involve computing $\nabla f(\cdot)$ and $\operatorname{prox}_{\gamma g}(\cdot)$ once.
#ManUp steps	Number of Riemannian steps, each involve computing $\operatorname{grad} F(\cdot)$ once and $\operatorname{Hess} F(\cdot)[\cdot]$ multiple times (one per Conjugate Gradient iteration).
#$\operatorname{Hess} F(\cdot)[\cdot]$	Number of Riemannian Hessian-vector products, approximates the effort spent in manifold updates since algorithm started.
#f	Number of calls to $f(\cdot)$, one per iteration, some for the Riemannian linesearch, some for the backtracking estimation of the Lipchitz constant of ∇f.
#g	Number of calls to $g(\cdot)$, one per iteration, some for the Riemannian linesearch.

The proximal gradient updates, present in all methods, include a backtracking procedure that maintains an estimate of the Lipschitz constant of ∇f, so that the proximal gradient stepsize is taken as the inverse of that estimate. The Conjugate Gradient used to solve (Newton equation) and (Inexact Newton equation) follows (Boumal, 2020, Alg. 6.2); it is stopped when the (in)exactness criterion is met, or after 50 iterations for the logistic problem and 150 for the trace-norm one, or when the inner direction d makes the ratio $\langle \operatorname{Hess} F(x_k)[d], d\rangle / \|d\|$ small. The manifold updates are completed by a backtracking linesearch started from unit stepsize, a direct implementation of (Dennis Jr and Schnabel, 1996, Alg. 6.3.1).

Computing the Riemannian gradient of F requires computing $\operatorname{grad} f(x_k)$, from $\nabla f(x_k)$ and (2.16), and $\operatorname{grad} g(x_k)$ which expression can be derived explicitly. The Riemannian Hessian-vector product is computed similarly, using $\nabla^2 F(x_k)[\eta_k]$ with (2.17) and the derived formulas for $\operatorname{Hess} g(x_k)[\eta_k]$. Doing so, only Hessian-vector products are computed and no hessian matrix is formed. In practice, we observe that the time required for computing $\operatorname{grad} f(x_k)$ and $\operatorname{Hess} f(x_k)[\eta_k]$ is about twice and four times that of computing $f(x_k)$.

The representation of iterates plays an important role in the method global efficiency, especially when the global space $\mathbb{R}^n$ is high-dimensional and the current manifold $\mathcal{M}$ low-dimensional. It seems attractive in that case to represent and manipulate data in a low-dimensional form, such as the truncated SVD representation of matrices.

6.3.1 Two-dimensional nonsmooth example

We consider the piecewise quadratic problem of (Lewis and Wylie, 2019):

$$\min_{x\in\mathbb{R}^2} 2x_1^2 + x_2^2 + |x_1^2 - x_2|. \tag{6.7}$$

The objective function is partly-smooth relative to the parabola $\mathcal{M} \triangleq \{x : x_2 = x_1^2\}$, for which an expression for the tangent space, the orthogonal projection on tangent space, a second-order retraction and conversion from Euclidean gradients and hessian-vector products to Riemannian ones are readily available. We detail here the different oracles of $f(x) \triangleq 2x_1^2 + x_2^2$ and $g(x) \triangleq |x_1^2 - x_2|$:

- *proximity operator:* For $\gamma < 1/2$, there holds

$$\operatorname{prox}_{\gamma g}(x) = \begin{cases} (\frac{x_1}{1+2\gamma}, x_2 + \gamma) & \text{if } x_2 \leq \frac{x_1^2}{(1+2\gamma)^2} - \gamma \\ (\frac{x_1}{1+4\gamma t-2\gamma}, x_2 + 2\gamma t - \gamma) & \text{if } \frac{x_1^2}{(1+2\gamma)^2} - \gamma \leq x_2 \leq \frac{x_1^2}{(1-2\gamma)^2} + \gamma \\ (\frac{x_1}{1-2\gamma}, x_2 - \gamma) & \text{if } \frac{x_1^2}{(1-2\gamma)^2} + \gamma \leq x_2 \end{cases}$$

  where t solves $x_2^2 + (-2\gamma t + \gamma - x_2)(1 + 4\gamma t - 2\gamma)^2 = 0$.

- *Riemannian gradient and hessian:* Since g is identically null on $\mathcal{M}$, for any point $(x, \eta) \in T\mathcal{B}$,

$$\operatorname{grad} g(x) = 0 \qquad \text{and} \qquad \operatorname{Hess} g(x)[\eta] = 0.$$

  Besides, Euclidean gradient and Hessian-vector product are converted to Riemannian ones using equations (2.16) and (2.17):

$$\operatorname{grad} f(x) = \operatorname{proj}_x(\nabla f(x))$$
$$\operatorname{Hess} f(x)[\eta] = \operatorname{proj}_x\left(\nabla^2 f(x)[\eta] - \begin{pmatrix} 2\eta_1 \\ 0 \end{pmatrix} \left\langle \nabla f(x), \begin{pmatrix} 2x_1 \\ -1 \end{pmatrix} \right\rangle \frac{1}{1 + 4x_1^2}\right),$$

  and the orthogonal projection onto $T_x\mathcal{M}$ writes

$$\operatorname{proj}_x(d) = d - \left\langle d, \begin{pmatrix} 2x_1 \\ -1 \end{pmatrix} \right\rangle \frac{1}{1 + 4x_1^2} \begin{pmatrix} 2x_1 \\ -1 \end{pmatrix}$$

We run the proximal gradient, its accelerated counterpart and Algorithm 6.1 with the Newton update Algorithm 6.2. The proximal gradient steps of all algorithms have a constant step-size $\gamma = 0.05$, all algorithms are started from point $(2, 3)$, and converge to the minimizer of the function $(0, 0)$.

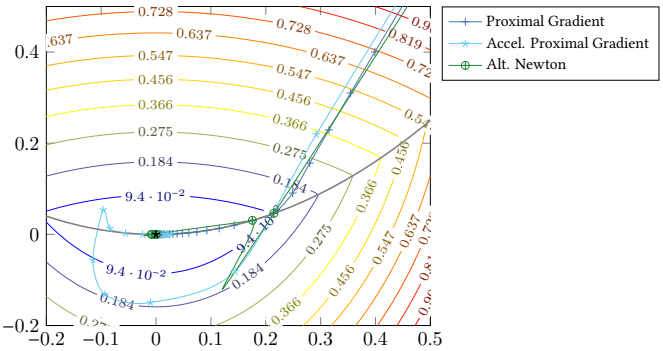
Observations The iterates are displayed in Fig. 6.2. The first Proximal Gradient step (not visible on the figure) yields an iterate lying on the parabola. While the (accelerated) proximal gradient iterates leave the parabola to reach it later (a typical behavior as observed in Chapter 4), the Alt. Newton iterates remains on the parabola until convergence. The quadratic convergence behavior appears clearly as two Newton manifold updates bring suboptimality below 10^{-3}, and one additional step gets below 10^{-12}. The Proximal Gradient iterates reach the parabola in finite time, and then converge linearly towards $x^\star$ on the parabola. The Accelerated Proximal Gradient iterates feature two known negative properties: they "overshoot" the manifold to be identified twice before reaching it, and they oscillate around $x^\star$ in the parabola (see Section 4.2.2).

6.3.2 ℓ_1-regularized logistic problem

We now turn to the ℓ_1-regularized logistic problem:

$$\min_{x \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \log(b_i \sigma(\langle a_i, x \rangle)) + \lambda \|x\|_1, \tag{6.8}$$

where $a_i \in \mathbb{R}^n$ and $b_i \in \{-1, 1\}$ for all $i = 1, .., m$, λ denotes a positive scalar and σ the sigmoid map $x \mapsto 1/(1 + \exp(-x))$. The nonsmooth part is the ℓ_1-norm $g(x) = \lambda \|x\|_1$ (see Example 2.24).



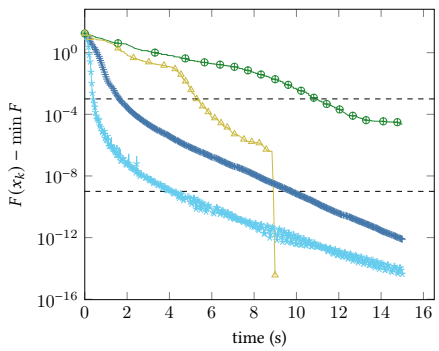
Algorithm	Tolerance	$F(x_k) - \min F$	#prox. grad. steps	#ManUp steps	#Hess $F(\cdot)[\cdot]$	$\#f$	$\#g$
Prox. Gradient	$1 \cdot 10^{-3}$	0.0007736370745816156	29	–	–	30	30
Prox. Gradient	$1 \cdot 10^{-9}$	7.594823106800896e-10	60	–	–	61	61
Accel. Prox. Gradient	$1 \cdot 10^{-3}$	0.0009626158394158507	16	–	–	17	17
Accel. Prox. Gradient	$1 \cdot 10^{-9}$	5.184441449855132e-10	63	–	–	64	64
Alt. Newton	$1 \cdot 10^{-3}$	0.00014946622308458022	2	2	10	7	7
Alt. Newton	$1 \cdot 10^{-9}$	8.748444107374269e-13	3	3	15	10	10

Figure 6.2: Example of Eq. (6.7)

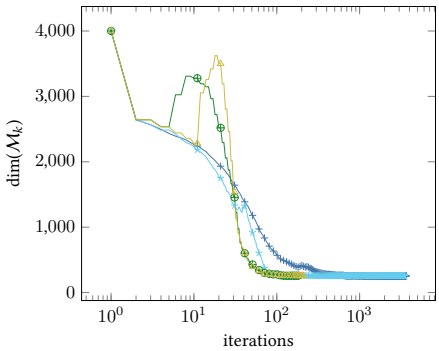
We consider an instance of (6.8) where $n = 4000$, $m = 400$, $\lambda = 10^{-2}$ and the final manifold has dimension 249. The coefficients of a are drawn independently following a normal law. The measurements b are generated as follows: a sparse random source signal s is generated, where each coordinate either is drawn following a normal law or is set to 0, with probability $1/2$. Measurement b_i is 1 with probability $(1+\sigma(\langle \mathrm{a}_i, s\rangle))/2$, and -1 otherwise.

Observations The experiments are presented in Fig. 6.3. The optimal manifold is identified roughly around iteration 200 for all methods except for Proximal Gradient, which needs about 1000 iterations. The two baselines Proximal Gradient and its accelerated version show linear convergence, with a better rate for the non accelerated version once the final manifold is reached. Alt. Truncated Newton shows superlinear acceleration, while Alt. Newton fails to converge in the given time budget. This fact, along with the counts of Hessian-vector products, shows the interest of the inexact solve of Newton equation performed by the Truncated Newton procedure Algorithm 6.3 as opposed to the exact solve performed by the Newton procedure Algorithm 6.2.

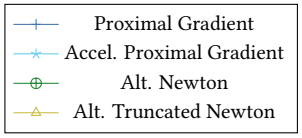
As iterations grow, the (Accelerated) Proximal Gradient identifies manifolds of decreasing dimension in a roughly monotonical way. Alt. Truncated Newton behaves differently: after identifying monotonically manifolds of dimension lower than 2000, the dimension of the current manifold jumps to about 3000 for about 10 iterations, to finally reach quickly the final manifold. We believe that this partial loss of identified structure is caused by iterates getting close to a non differentiable point of F *on the current manifold*, *e.g.* having one non-null but very small coordinate. There, the second-order Taylor extension is valid on a small set, broadly speaking only up to the non-differentiability point; however it may lead to a Newton step that lies outside that set, thus driving the iterate away from more structured points instead of identifying them. The same behavior occurs for Alt. Newton. This difficulty can be related to the well-known problem of constraint activation in nonlinear programming. Despite this behavior, Algorithm 6.1 retains a good rate overall.



(a) Suboptimality vs time (s)



(b) Manifold dimension vs iteration



Algorithm	Tolerance	$F(x_k) - \min F$	#prox. grad. steps	#ManUp steps	#Hess $F(\cdot)[\cdot]$	$\#f$	$\#g$
Prox. Gradient	$1 \cdot 10^{-3}$	0.0009963198229036019	357	–	–	779	358
Prox. Gradient	$1 \cdot 10^{-9}$	9.965078207052613e-10	2306	–	–	4677	2307
Accel. Prox. Gradient	$1 \cdot 10^{-3}$	0.0009257766624239938	90	–	–	246	91
Accel. Prox. Gradient	$1 \cdot 10^{-9}$	9.899422392933843e-10	953	–	–	1972	954
Alt. Newton	$1 \cdot 10^{-3}$	0.0009759231753842523	62	61	6303	556	427
Alt. Newton	$1 \cdot 10^{-9}$	–	–	–	–	–	–
Alt. Truncated Newton	$1 \cdot 10^{-3}$	0.0009557819627238895	51	50	2616	437	321
Alt. Truncated Newton	$1 \cdot 10^{-9}$	3.774758283725532e-15	105	105	5091	742	572

Figure 6.3: Logistic-ℓ_1 problem

6.3.3 *Trace-norm regularized problem*

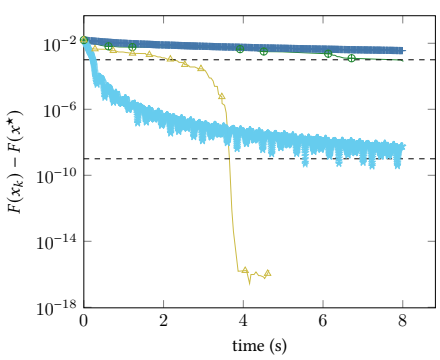
We finally consider a matrix regression problem:

$$\min_{X \in \mathbb{R}^{n_1 \times n_2}} \frac{1}{2} \sum_{i=1}^{m} \left(\langle A_i, X \rangle - y_i \right)^2 + \lambda \|X\|_*, \tag{6.9}$$

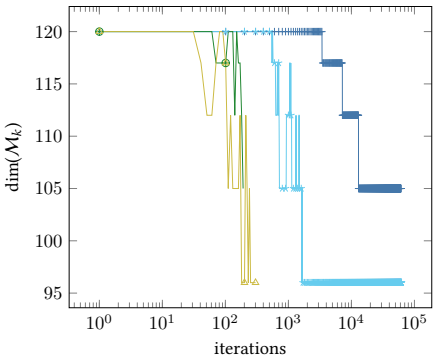
where $A_i \in \mathbb{R}^{n_1 \times n_2}$ for $i = 1, .., m$, $y \in \mathbb{R}^m$ and λ denotes a positive scalar. The nonsmooth part is the nuclear norm $g(x) = \lambda \|x\|_*$ (see Example 2.28).

We consider an instance of (6.9) where $n_1 = 10$, $n_2 = 12$, $m = 60$, $\lambda = 1 \cdot 10^{-2}$ and the final manifold is that of matrices of rank 6. The coefficients of each matrix A_i are drawn independently following a normal law. The measurements y are generated as follows: a sparse random source signal s is generated as the projection of a matrix which coordinates are drawn independently following a normal law; the measurement y_i is then $\langle A_i, s \rangle + \xi_i$, where ξ_i follows a centered gaussian law with variance 0.01^2.

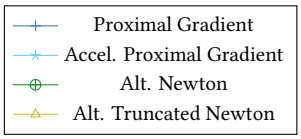
Observations The experiments are presented in Fig. 6.4. We see on Fig. 6.4a that the Proximal Gradient algorithm and its accelerated version converge sublinearly, which is to be related to the lack of strong convexity of the objective problem. Alt. Truncated Newton converges superlinearly, and shows the interest of the Newtonian acceleration. Figure 6.4b shows that the Proximal Gradient does not reach the final optimal manifold within the budget of iterations; similarly for the Newton method, within the budget of time.



(a) Suboptimality vs time (s)



(b) Manifold dimension vs iteration



Algorithm	Tolerance	$F(x_k) - \min F$	#prox. grad. steps	#ManUp steps	#Hess $F(\cdot)[\cdot]$	$\#f$	$\#g$
Prox. Gradient	$1 \cdot 10^{-3}$	–	–	–	–	–	–
Prox. Gradient	$1 \cdot 10^{-9}$	–	–	–	–	–	–
Accel. Prox. Gradient	$1 \cdot 10^{-3}$	0.00099894916795637	1489	–	–	3073	1490
Accel. Prox. Gradient	$1 \cdot 10^{-9}$	9.858174276899945e-10	43283	–	–	86661	43284
Alt. Newton	$1 \cdot 10^{-3}$	0.0009833250032105778	93	93	28063	873	687
Alt. Newton	$1 \cdot 10^{-9}$	–	–	–	–	–	–
Alt. Truncated Newton	$1 \cdot 10^{-3}$	0.0009695009931029591	76	76	16342	738	568
Alt. Truncated Newton	$1 \cdot 10^{-9}$	2.2716245551279712e-11	128	128	27786	1101	879

Figure 6.4: Trace-norm problem

Similarly to the logistic problem, Algorithm 6.1 shows some oscillation between manifolds upon identification of the optimal one. We believe that, here as well, that this is due to iterates having positive but near-zero singular values. In this case, the oscillation behavior seems less harmful, which may be related to the fact that manifolds are nested in one another.

6.4 Concluding remarks

In this chapter, we saw that it is possible to plug very efficient Riemannian steps on the manifolds identified by a proximal gradient algorithm. The key ingredients in the analysis to link the proximal gradient with Riemannian steps are once again based on the notions of qualification condition and partial smoothness described in Chapter 3.

Without assuming convexity nor a prior knowledge of the final structure, the proposed method directly adapts to the uncovered Riemannian structure and eventually benefit from a super-linear rate when the critical points are qualified. Numerically, the identified Riemannian structure and methods are efficiently implementable for our cases of interest. Furthermore, we notice that even though we are not always sure that the reached critical points are qualified, our methods still provide an interesting acceleration, this is intuitively due to being able to capture the better conditioning of the problem on lower-dimensional manifolds.

This exactly resonates with the general objective of this part. We are able to harness

the structural information explicitly brought by a proximal method to a numerical advantage by using Riemannian (inexact) Newton method on the identified manifold; all of this while being completely adaptive to the structure while benefiting from the same final superlinear rates as if the final manifold is known!

PART B

DISTRIBUTED STRUCTURE & ASYNCHRONY

In this part, we consider the situation an optimization problem is split over several machines. More precisely, we place ourselves in a context where M workers/machines/agents are each given a function $F^i : \mathbb{R}^n \to \overline{\mathbb{R}}$ and can exchange information (*i.e.* points, gradients, function values, etc.) with a coordinator/master/central authority. We are thus in the centralized/master-worker setting, not in the decentralized case where machines would directly exchange with each other.[46]

Mathematically, a distributed optimization problem can usually be written as

$$\min_{x \in \mathbb{R}^n} F(x) \;:= \sum_{i=1}^{M} F^i(x) \tag{$\mathcal{P}_B$}$$

where the *global* objective function F is the sum of the workers functions (F^i). The *sum* structure is clearly the most widespread in the literature and is central to the forthcoming chapters.[47] This kind of structure arises naturally when minimizing some error over data that is scattered among the workers. Alternatively, it can be obtained by splitting a function into M chunks and solving the Lagrangian dual problem.

[46] The latter setting is also interesting to me and represented a significant part of the contributions of my PhD thesis. However, this topic is less representative of my recent research, to the exception of (Iutzeler, 2017; Iutzeler and Condat, 2018).

[47] Nevertheless, different distributed structures have separate interests both practically and theoretically. The sum could for instance be replaced by a max, or a median, etc. depending on practical applications as discussed in the perspectives Chapter 9.

Motivation: Adapting algorithms to the computing setup

Distributed methods for solving ($\mathcal{P}_B$) have been studied for decades and particularly since the pioneering works of John Tsitsiklis and Dimitri Bertsekas (Bertsekas and Tsitsiklis, 1989; Tsitsiklis et al., 1986; Tsitsiklis, 1984). These distributed algorithms have two main advantages over standard methods. First, if each F^i depends on local data at machine i, it may be preferable to directly query the machines for points, gradients, or functional values rather than regrouping all the data in one place; be it for practical or legal reasons. Second, with M machines the computational power is increased. Even though a linear speed-up seems hard to achieve due to the potential idle times, communication latencies, or heterogeneity of the system, we can hope to accelerate the optimization process.

These advantages explain the practical success of these methods from multiprocessor machines (which were the primary target in the 1980's (Tsitsiklis et al., 1986)) to sensor networks (Rabbat and Nowak, 2004) and more recently clusters of machines (Boyd et al., 2011; Hall et al., 2010), massive multicore units (Niu et al., 2011), and globally networked systems including the recent federated learning framework (Kairouz et al., 2019; Konečnỳ et al., 2016).

Distributed Optimization Methods

The conception of methods for solving ($\mathcal{P}_{\mathrm{B}}$) depends heavily on i) if the objective is shared by all workers or not; ii) the communication schemes allowed between the coordinator and the workers; and iii) the worker functions' properties. Here, far from addressing all possibilities, we choose some specific settings to focus our work. The general motivation for our choices is guided by practical implementations, curiosity, and applications in data science.

Shared vs. Scattered tasks

When the objective is shared by all workers (*i.e.* if $F^i = \tilde{F}$ for all i), this means that all participants either have the same information or access to a shared memory where the problem data is stored. This situation arises when parallelizing loops or coordinate descent methods over several computing units, see *e.g.* (Bareilles et al., 2020b; Ma et al., 2015; Peng et al., 2016). The main challenges in this setup is to deal with potential inconsistent cache read/write and synchronization to offer better performances.

In this part, we will consider a setting where each worker has its own local objective (typically associated with its own data). This solution is often cheaper and more scalable than shared-memory systems. Recent computing architectures thus often rely on many heterogeneous nodes rather than a few very powerful ones. These changes call for adapted algorithms with a special attention to communications.

Communication Scheme

In this part, we adopt a radical stance on the communication scheme:

the workers exchange with the coordinator asynchronously
without any control nor knowledge about the delays.

In order for the setup to make sense, we will obviously assume that all machines exchange within a finite time, but we will not assume the knowledge or the existence of a bound on this time. This places us in the *totally asynchronous* setup following the terminology of (Bertsekas and Tsitsiklis, 1989, Chap. 6.1). The main advantage of this approach is to able to carry on computation without waiting for slower machines: machines tirelessly perform computations based on potentially outdated versions of the global state while a coordinator gathers the inputs into an efficient update. In traditional synchronous algorithms, latency, bandwidth limits, and unexpected drains on resources that delay the update of machines would cause the entire system to wait. By eliminating the idle times, asynchronous algorithms can be much faster than traditional ones; see *e.g.* (Hannah and Yin, 2017).

This choice is motivated by implementation possibilities. In Python or C/C++, it is fairly easy to implement asynchronous exchanges between one machines and others using the non-blocking *send/receive* of the MPI standard. In Julia, the module Distributed is part of the standard library and provides an easy way to add *processors* (*e.g.* distant machines) that can communicate through independent *Channel* objects. These possibility enable us to implement algorithms on multiples machines without being restricted to synchronous Map/Reduce operations.

With the asynchronous implementations mentioned above, the system performs the best it can, with few idle times, but the user does not have any control on the order or the time in which a machine will respond. In addition, even if the machines

response time can be limited, the delay that appears mathematically is the number of updates in the system since the last exchange of the machine. This means that if some machine is very fast, it will increase the system's delays. Finally, delays bounds are generally very pessimistic since the periods of lag or bottlenecks are often transient.

Workers' local functions

As in the single machine case, different algorithms shall be employed when the functions are smooth, possess easily computable proximity operators, or can be merely queried for subgradients. Since the machines possess computing capacities, one could question whether to ask for a basic oracle (*e.g.* a gradient), a quick approximate one (*e.g.* a stochastic gradient), or a more precise one (*e.g.* an inexact proximal step).

In this part, we focus one the basic ones: gradients, subgradients, proximity operator (if explicit). The reason for that is that cheaper ones may lead to too frequent communications and more precise ones may lead to wasted computational power. Out of simplicity. This leads to easier implementation and is probably more in line with practical solutions.

Objectives of the part

The worker functions' properties will separate the two chapters of this part. This is due to the very different nature of the proofs techniques for the cases distributed. Nevertheless, for both chapters, the central question is

How to efficiently harness asynchronous oracles?

We show in this part the importance of the choice of the combination for the resilience of the algorithm to heterogeneous delays. Our viewpoint is to handle explicitly the delays as opposed to treating them as noise. This has two general consequences: i) the proposed methods' parameters do not depend on the delays; and ii) the proofs have a striking common point: an interesting proxy for a *global iteration* is when all machines have exchanged *twice*.[48]

[48] Intuitively, all machines have to exchange once to get a sufficiently new point and a second time so that the master gets a feedback on these new points.

In Chapter 7, we will consider general nonsmooth functions for which the workers can produce subgradient oracles. In this type of problems, commonly arising from Lagrangian relaxations of difficult/non-convex problems, bundle methods are the techniques of choice. We show here how to construct global lower-models from asynchronous local subgradient information and how to produce bundle methods exploiting these models.

In Chapter 8, we focus on a distributed version of the composite setting of Part A. Driven by applications in Machine Learning, each machine has a smooth loss function depending on local data and a global nonsmooth regularization. We exhibit an efficient way of extending the proximal gradient algorithm. Furthermore, we use identification to further reduce the cost of exchanges.

7 Asynchronous Level Bundle

There are two schools of thought about the resilience of time.
The first is that time is highly volatile, with every small event
altering the possible outcome of the earth's future.
The other view is that time is rigid, and no matter how hard you
try, it will always spring back toward a determined present.
Myself, I do not worry about such trivialities. I simply sell ties to
anyone who wants to buy one...
Jasper Fforde – *The Eyre Affair* (2001)

We consider nonsmooth convex optimization problems with additive structure featuring independent oracles (black-boxes) working in parallel. Existing methods for solving these distributed problems in a general form are synchronous, in the sense that they wait for the responses of all the oracles before performing a new iteration. Here, we propose level bundle methods handling asynchronous oracles. These methods are based on the idea of disaggregated bundle and require original upper-models to deal with asynchronicity.

This chapter is based on the following publication:

- F. Iutzeler, J. Malick, and W. de Oliveira : Asynchronous level bundle methods, Mathematical Programming, vol. 184, pp. 319-348, 2020.

We consider convex optimization problems of the form

$$\min_{x\in\mathcal{X}} F(x) := \sum_{i=1}^{M} F^i(x) \qquad (\mathcal{P}_{\text{B}-7})$$

where the constraint set $\mathcal{X}$ is compact convex, and the functions $F^i : \mathbb{R}^n \to \mathbb{R}$ (for all $i = 1, .., M$) are convex and possibly nonsmooth.[49]

The nonsmoothness of the F^i typically come from the fact that they are themselves the results of inner optimization problems, as in Lagrangian relaxation (Lemaréchal, 2001), stochastic optimization (Shapiro et al., 2009), or in Benders decomposition (Geoffrion, 1972). In such cases, the functions F^i are known only implicitly through oracles providing values and subgradients (or approximations of them) at a given point.

The so-called *bundle methods* are a family of nonsmooth optimization algorithms adapted to the resolution of such problems. Using zero-th and first order oracle information, these methods construct cutting-plane approximations of the objective function together with quadratic stabilization techniques. Bundle methods can be traced back

[49] In short, we consider ($\mathcal{P}_{\text{B}}$) with an additional compact constraint set, mainly to avoid degeneracy issues in bundle models.

to (Lemaréchal, 1975); we refer to the textbook (Hiriart-Urruty and Lemaréchal, 1993) and the recent surveys (Frangioni, 2020; Oliveira and Sagastizábal, 2014) for relevant references. Real-life applications of such methods are numerous, ranging from combinatorial problems (Briant et al., 2008) to energy optimization (Bruno et al., 2017; Sagastizábal, 2012).

In line with the part's general problem, we consider here the situation where ($\mathcal{P}_{\mathrm{B}-7}$) is scattered over several machines: each function F^i corresponds to one machine and associated local data and computing abilities. This scattering can be due to the privacy of the data, its prohibitive size, or the prohibitive computing power required to deal with the full problem. In our context, the functions F^i are nonsmooth and only known through oracles. Moreover, in many applications, some of these oracles require considerably more time than others. This motivated us to turn our attention to *asynchronous* algorithms.

Related literature

Though asynchrony is extremely important for efficiency and resilience of distributed computing, no asynchronous algorithm exists for solving ($\mathcal{P}_{\mathrm{B}-7}$) in the above general distributed setting. For example, big problems in stochastic optimization (where uncertainty is for example due to intermittent renewable energy sources) are heavily structured, often amenable to parallel computing by standard decomposition schemes (*e.g.* by scenarios (Bruno et al., 2017; Rockafellar and Wets, 1991), by production units (Dubost et al., 2005), or even both (van Ackooij and Malick, 2015)). However, existing optimization algorithms exploiting this decomposability are all synchronous, expect for the recent (Kim et al., 2019) (see also references therein) dealing with a specific polyhedral problem.

Convincingly, bundle methods are particularly well-suited for asynchronous generalizations: outdated information provided by late machines could indeed be considered as inexact linearizations (see (Malick et al., 2017), the review (de Oliveira and Solodov, 2018), and references therein). However, to our knowledge, there is no asynchronous version of bundle methods for the general distributed setting. Indeed, there does exist an asynchronous (proximal) bundle method (Fischer and Helmberg, 2014) but it is tailored for the case where M is large and each component depends only on some of the variables. Moreover its implementation and its analysis are intricate and do not follow the usual rationale of the literature. We can also mention the asynchronous bundle (trust-region) method of (Kim et al., 2019), designed for dual decomposition of two-stage stochastic mixed-integer problems. This algorithm requires to eventually call all the oracles for all the iterates, which we want avoid in our general situation. Another related work is the incremental (proximal) bundle algorithm of (Wim van Ackooij, 2016), that could serve as a basis for an asynchronous generalization. However, this generalization is not provided or discussed in (Wim van Ackooij, 2016).

Here, we propose, analyze, and illustrate the first asynchronous bundle method adapted to the general centralized distributed setting, encompassing computer clusters or mobile agents, described previously. We will not build on the two aforementioned *proximal* bundle methods of (Fischer and Helmberg, 2014; Wim van Ackooij, 2016), but rather investigate *level* bundle methods (Kiwiel, 1995; Lemaréchal et al., 1995).

In contrast with proximal methods where the iterates' moves can be small even with reasonably rich cutting-planes, level bundle methods fully benefit from collected asynchronous information: richer cutting-plane models would tend to generate useful lower bounds, so that the level set would better approximate the solution set, and the

next iterate would better approximate an optimal solution. Such behavior is discussed in the related context of uncontrolled inexact linearizations; see the comparisons between proximal and level methods in Section 4 of (Malick et al., 2017).

Though asynchronous linearizations of the functions and associated lower-bounds can be well exploited in cutting-plane approximations, existing (level) bundle methods cannot be readily extended in an asynchronous setting because of the lack of upper-bounds: the values of the objective function are never available, since the oracles have no reason to be all called upon the same point. The main algorithmic difficulty is thus to find and manage upper-bounds within asynchronous bundle methods. This is quite specific to bundle methods which rely on estimates of functions values or upper-bounds on the optimal values to compute ad-hoc iterates.

7.1 Level bundle methods: recalls & disaggregated version

This section reviews the main ideas about (synchronous) level bundle methods. In this chapter, we will focus on nonsmooth convex problems.

Assumption 7.1. For all $i = 1, .., M$, the functions $F^i : \mathbb{R}^n \to \mathbb{R}$ (for all $i = 1, .., M$) are convex and $\mathcal{X}$ is compact convex.

In Section 7.1.1, we recall the classical algorithm with the associated notation. Then, we describe in Section 7.1.2 a disaggregated variant exploiting the decomposability of the objective function of ($\mathcal{P}_{B-7}$). The use of disaggregate models is well-known for proximal bundle methods, but not fully investigated for level bundle methods: we develop here the useful material for the asynchronous algorithms of the next sections.

7.1.1 Main ingredients of level bundle methods

We briefly present the classical scheme of level bundle methods, dating back to (Kiwiel, 1995; Lemaréchal et al., 1995), to minimize over $\mathcal{X}$ the function F only known through an exact oracle. While presenting the algorithm and its main features, we introduce standard notation and terminology of bundle methods; see *e.g.* the textbook (Hiriart-Urruty and Lemaréchal, 1993).

Bundle algorithms produces a sequence of feasible points $(x_k) \subset \mathcal{X}$ fo which the oracle information $F(x_k)$ and $v_k \in \partial F(x_k)$ is assumed to be available. With such information, these methods create linearizations of the form $F(x_k) + \langle v_k, \cdot - x_k \rangle \leq F(\cdot)$. Such linearizations are used to create a cutting-plane model for F at iteration k:

$$\check{F}_k(x) := \max_{j \in J_k} \{F(x_j) + \langle v_j, x - x_j \rangle\} \quad \leq \quad F(x) \tag{7.1}$$

where $J_k \subset \{1, 2, .., k\}$ is a set of indices of points at which the oracle was called.

The use of theses piecewise linear model is at the heart of bundle methods and in particular of *level* bundle methods. Their principal feature is the use of a *level set* which contains the points in $\mathcal{X}$ at which the cutting-plane model is smaller than some *level parameter* . For an iteration k, we define the level set $\mathbb{X}_k$ of $\check{F}_k$ associated with the level parameter $F_k^{\text{lev}} \in \mathbb{R}$

$$\mathbb{X}_k := \{x \in \mathcal{X} : \check{F}_k(x) \leq F_k^{\text{lev}}\} \quad \supset \quad \{x \in \mathcal{X} : F(x) \leq F_k^{\text{lev}}\}.$$

This level set defines a region of the feasible set in which the next iterate will be chosen: $x_{k+1} \in \mathbb{X}_k$. It can also provide a lower bound F_k^{low} on $F^\star$. Indeed, whenever $\mathbb{X}_k = \emptyset$ one can take $F_{k+1}^{\text{low}} = F_k^{\text{lev}}$ as, in this case, $F^\star \geq F_k^{\text{lev}}$.

A common rule to choose the next iterate is by projecting a certain stability center $\hat{x}_k \in (x_k)$ onto $\mathbb{X}_k$, that is taking x_{k+1} as the optimization sub-problem

$$\min_{x \in \mathbb{X}_k} \frac{1}{2}\|x - \hat{x}_k\|^2. \tag{7.2}$$

When $\mathcal{X}$ is a polyhedral set, computing the next iterate consists in solving a mere convex quadratic optimization problem. This is also the case when $\mathcal{X}$ is a Euclidean ball, as pointed out in (de Oliveira, 2017). Since the sequence (x_k) is available, an upper bound for the optimal value $F^\star$ is just $F_k^{\text{up}} := \min_{j \in \{1,..,k\}} F(x_j)$.

Hence, at each iteration k, using the level set $\mathbb{X}_k$, we can compute an upper and a lower bound on the optimal value. We can then define the gap

$$\Delta_k := F_k^{\text{up}} - F_k^{\text{low}}$$

which gives a natural stopping test $\Delta_k \leq \text{tol}_\Delta$, for some stopping tolerance $\text{tol}_\Delta \geq 0$. Indeed, $\Delta_k \leq \text{tol}_\Delta$ implies that the best point generated by the algorithm so far $x_{\text{best}} = \text{argmin}\{F(x) : x \in \{x_1, .., x_k\}\}$ verifies $F(x_{\text{best}}) - F^\star \leq \text{tol}_\Delta$ and thus has a suboptimality bounded by tol_Δ.

Finally, we need to update the level parameter F_k^{lev}. Intuitively, it has to be taken between F_k^{low} and F_k^{up} (otherwise $\mathbb{X}_k$ would be empty or too big respectively). A natural choice is thus to take a convex combination of both:

$$F_k^{\text{lev}} := \alpha F_k^{\text{low}} + (1-\alpha) F_k^{\text{up}} = F_k^{\text{up}} - \alpha\Delta_k, \qquad \text{with } \alpha \in (0,1). \tag{7.3}$$

It remains to find a policy for updating the stability center $\hat{x}_k$ used in (7.2). It could be updated each time a new best point is found, but for more stability it seems preferable to update it only when the gap has been reduce by a factor α (the same as in (7.3)) since the last update of the stability center. The obtained algorithm is displayed in Algorithm 7.1 and essentially corresponds to the method presented in (Kiwiel, 1995).

Remark 7.2 (About convergence). The convergence of the standard level bundle method presented in Algorithm 7.1 follows Theorem 3.5 in (Kiwiel, 1995). The proof uses an argument which does not hold in the asynchronous setting: $x_{k+1} \in \mathbb{X}_k$ implies that $F(x_k) + \langle v_k, x_{k+1} - x_k\rangle \leq F_k^{\text{lev}}$ for $k \in J_k$, that in turn yields

$$\|x_{k+1} - x_k\| \geq \frac{F(x_k) - F_k^{\text{lev}}}{\|v_k\|} \geq \frac{\alpha\Delta_k}{\Lambda}, \tag{7.4}$$

where the bound $\|v_k\| \leq \Lambda$ can be guaranteed by compactness of $\mathcal{X}$ and convexity of F. Since this inequality requires the value of $F(x_k)$, it is not guaranteed anymore in the asynchronous setting. We will pay special attention to this technical point in the algorithms of Section 7.3. ◄

Algorithm 7.1 Standard level bundle method

Choose $x_1 \in \mathcal{X}$ and set $\hat{x}_1 = x_1$
Define $F_1^{\text{up}} = +\infty$, $J_0 = \emptyset$ and $\hat{\Delta} = +\infty$
Choose a stopping tolerance $\text{tol}_\Delta \geq 0$, a parameter $\alpha \in (0,1)$ and a finite $F_1^{\text{low}} \leq F^\star$
Send x_1 to the oracle
for $k = 1, 2, \ldots$ **do**

STEP 1: RECEIVE INFORMATION FROM ORACLE

  Receive $(F(x_k), v_k)$ from the oracle
  Set $J_k = J_{k-1} \cup \{k\}$
  if $F(x_k) < F_k^{\text{up}}$ **then**
    Update $F_k^{\text{up}} = F(x_k)$ and $x_{\text{best}} = x_k$ ▷ update upper bound
  end if

STEP 2: TEST OPTIMALITY AND SUFFICIENT DECREASE

  Set $\Delta_k = F_k^{\text{up}} - F_k^{\text{low}}$
  if $\Delta_k \leq \text{tol}_\Delta$ **then**
    Return x_{best} and F_k^{up}
  end if
  if $\Delta_k \leq \alpha\hat{\Delta}$ **then**
    Set $\hat{x}_k = x_{\text{best}}$, set $\hat{\Delta} = \Delta_k$, and possibly reduce J_k ▷ critical iterate
  end if

STEP 3: COMPUTE NEXT ITERATE

  Set $F_k^{\text{lev}} = F_k^{\text{up}} - \alpha\Delta_k$. Run a quadratic solver on problem (7.2).
  if (7.2) is feasible **then**
    Get the new iterate $x_{k+1} \in \mathbb{X}_k$. Update $F_{k+1}^{\text{low}} = F_k^{\text{low}}$ and $F_{k+1}^{\text{up}} = F_k^{\text{up}}$
  else
    Set $F_k^{\text{low}} = F_k^{\text{lev}}$ and go to Step 2 ▷ update lower bound
  end if

STEP 4: SEND BACK INFORMATION TO THE ORACLE

  Send x_{k+1} to the oracle
  Set $\hat{x}_{k+1} = \hat{x}_k$
end for

7.1.2 Disaggregated level set

In order to present a level bundle algorithm which exploits the additive structure of the objective function ($\mathcal{P}_{\text{B-7}}$), we first need to define a *disaggregated level set* able to incorporate asynchronous updates from several workers.

In our setup, M oracles can provide individual information $(F^i(x), v^i) \in \mathbb{R}^{1+n}$ with $v^i \in \partial F^i(x)$ for a query point $x \in \mathcal{X}$. We define $J_k^i \subset \{1, .., k\}$ as the index set of the points in the sequence (x_k) where the oracle i was called, *i.e.* such that $(F^i(x_j), v_j^i)_j$ is computed. The unique feature of our situation and the main technical point is that the intersection of the index sets $\cap_{i=1}^M J_k^i$ may contain only few elements, or even be empty. To exploit the additive structure of the objective, we can define individual cutting-plane models for each i at an iteration k:

$$\check{F}_k^i(x) := \max_{j \in J_k^i} \left\{ F^i(x_j) + \langle v_j^i, x - x_j \rangle \right\} \quad \leq \quad F^i(x).$$

Instead of approximating the objective function F by its aggregate cutting-plane model $\check{F}_k$ presented in (7.1), we approximate F by its disaggregated model $\sum_{i=1}^M \check{F}_k^i$. While the idea is standard for proximal bundle methods (see *e.g.* (Bacaud et al., 2001) for an application in electricity management and (Frangioni and Gorgone, 2014) for an application in network optimization), the use of disaggregated models has not been fully investigated for level methods: we are only aware of the level algorithm of

(Wolf et al., 2014) that employs a disaggregate model indirectly just to request exact information from on-demand accuracy oracles.

If $J_k \subset \cap_{i=1}^M J_k^i$, then the disaggregated model provides a better approximation for F than the aggregated model based on J_k

$$\check{F}_k(x) \quad \leq \quad \sum_{i=1}^{M} \check{F}_k^i(x) \quad \leq \quad F(x) \quad \forall x \in \mathbb{R}^n.$$

We can then replace in the level bundle algorithm the quadratic subproblem (7.2) by the disaggregated quadratic subproblem:

$$\min_{x \in \mathbb{X}_k^{\mathrm{d}}} \frac{1}{2}\|x - \hat{x}_k\|^2 \text{ with } \mathbb{X}_k^{\mathrm{d}} := \left\{ x \in \mathcal{X} : \sum_{i=1}^{M} \check{F}_k^i(x) \leq F_k^{\mathrm{lev}} \right\}, \tag{7.5}$$

where $\hat{x}_k$ is the stability center and $F_k^{\mathrm{lev}} \in (F_k^{\mathrm{low}}, F_k^{\mathrm{up}})$ is the level parameter. As in Section 7.1, if Problem (7.5) is infeasible, *i.e.* if $\mathbb{X}_k^{\mathrm{d}}$ is empty, then F_k^{lev} is a lower bound for $F^\star$.

Aside from offering better accuracy, the disaggregate model has another advantage in our context: it allows for partial update of the cutting-plane model using individual oracle evaluations, without calling all of the M oracles at the same point. This is an important feature that permits to handle oracles in an asynchronous manner.

We formalize in the next lemma the easy result stating that (7.5) can be cast as a standard quadratic problem. Furthermore, if the problem is infeasible, $\mathbb{X}_k^{\mathrm{d}} = \emptyset$ which triggers the lower bound update as in Section 7.1.

Lemma 7.3 (Disaggregated master sub-problem)**.** *Let Assumption 7.1 hold and assume $\mathbb{X}_k^{\mathrm{d}}$ is nonempty. Then, the unique solution of (7.5) can be obtained by solving the following quadratic optimization problem (in both variables x and r) and discarding the r-component of its the solution:*

$$\left\{ \begin{array}{lll} \min_{x,r} & \frac{1}{2}\|x - \hat{x}_k\|^2 & \\ \text{s.t.} & x \in \mathcal{X}, r \in \mathbb{R}^M & \\ & F^1(x_j) + \langle v_j^1, x - x_j \rangle \leq r^1 & \forall j \in J_k^1 \\ & \vdots & \vdots \\ & F^M(x_j) + \langle v_j^M, x - x_j \rangle \leq r^M & \forall j \in J_k^M \\ & \sum_{i=1}^M r^i \leq F_k^{\mathrm{lev}}. & \end{array} \right. \tag{7.6}$$

The size of the disaggregated master quadratic problem is larger (with M additional variables and M times more constraints), which increases the computing time to solve it. In the case of costly oracles, this additional time is usually negligible compared to the total oracles computing time and the gain in precision thanks to the disaggregated cutting-plane model.

7.2 Asynchronous level bundle by upper-bound estimation

In this section, we present and analyze the first asynchronous level bundle method, using the disaggregated master subproblem (7.6) to incorporate asynchronous linearizations from the oracles.

One iteration corresponds to the treatment by the coordinator of the information sent by one oracle. More precisely, at iteration k, the master receives the information

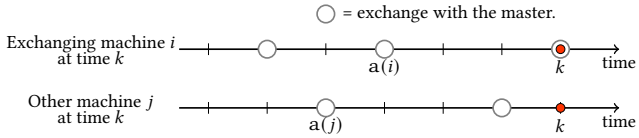


Figure 7.1: Notations for the delayed cuts added in our asynchronous bundle algorithms.

from an oracle, say i, updates the disaggregated model, generates a new point, and sends back a point to oracle i.

The asynchronicity in the algorithm implies that the oracles do not necessarily provide information on the last iterate but on a previous one, so that asynchronous bundle methods have to deal with delayed linearizations. As in the other chapters of this parts, we place ourselves in the totally asynchronous setting following the terminology of (Bertsekas and Tsitsiklis, 1989, Chap. 6); we assume that all oracles respond and are incorporated in finite time, but we do not need to upper bound theses response times.

In order to incorporate these delayed cuts, we denote by $\mathrm{a}(i)$ the iteration index of the *anterior* information provided by oracle i : at iteration k, the exchanging oracle i provides the linearization for F^i at the point denoted $x_{\mathrm{a}(i)}$ (see lines 6 and 7 of the Algorithm 7.2 and Fig. 7.1). Our finite response assumption then simply translates to $\mathrm{a}(i) \to \infty$ as $k \to \infty$ for all i.

7.2.1 Algorithm

Apart from the disaggregated level set and the asynchronous communication with the oracles, the main algorithmic difference between our asynchronous level bundle method and the standard level algorithm (Algorithm 7.1) is the management of upper-bounds F_k^{up}. The strategy presented here (and inspired by (Wim van Ackooij, 2016)) is to estimate upper bounds on $F^\star$ without evaluating all component functions F^i at the same point since this would require synchronizing the oracles very often. To this end, we make the assumption that we know an upper bound $\bar{\Lambda}^i$ on the Lipschitz constant Λ^i of each F^i for $i = 1, .., M$. In other words, we assume

$$|F^i(x) - F^i(u)| \leq \bar{\Lambda}^i \|x - u\| \qquad \text{for all } x, u \in \mathcal{X}. \tag{7.7}$$

The recent work (Wim van Ackooij, 2016) builds on the same assumption and proposes to bound $F^i(x)$ at a given point by solving an extra quadratic problem of size $|J_k^i| + 1$ depending on $\bar{\Lambda}^i$. Using this technique, we would obtain an upper-bound of $F(x)$ at the extra cost of solving $M - 1$ quadratic problems at each iteration.

We propose in Algorithm 7.2 a simpler procedure to compute upper bounds F_k^{up} (Note that our upper bound is computable only once all the oracles have responded once so that $\mathrm{a}(j)$ is well defined for all oracles.). Our strategy appears on line 23 of Algorithm 7.2 and is based on the following lemma establishing that one has an upper-bound on the full function at the current points if the points where the F^i are evaluated are "close enough". This yields a handy rule for updating F_k^{up} depending only on the distance (weighted by $\bar{\Lambda}^i$) between the current solution of the master sub-problem (7.6), x_{k+1}, and the last/anterior points on which the oracles responded, $x_{\mathrm{a}(i)}$ for $i = 1, .., M$.

Algorithm 7.2 Asynchronous level bundle method by upper-bounds estimation

Get $\bar{\Lambda}^i$ satisfying (7.7). Choose $x_1 \in \mathcal{X}$, and set $x_{\text{best}}^- = \hat{x}_1 = x_1$
Choose a tolerance $\text{tol}_\Lambda \geq 0$, a parameter $\alpha \in (0,1)$, and $F_1^{\text{up}} > F^\star + \text{tol}_\Lambda$ and $F_1^{\text{low}} \leq F^\star$
Set $\hat{\Delta} = \infty$ and ${J_0}^i = \emptyset$ for all $i = 1, .., M$
Send x_1 to the M oracles
for $k = 1, 2, \ldots$ **do**

Step 1: receive information from an oracle

  Receive from oracle i the oracle information on F^i at a previous iterate $x_{k'}$
  Set $\mathtt{a}(i) = k'$, store $(F^i(x_{k'}), v_{k'})$, and set $J_k^i = J_{k-1}^i \cup \{k'\}$

Step 2: test optimality and sufficient decrease

  Set $\Delta_k = F_k^{\text{up}} - F_k^{\text{low}}$
  if $\Delta_k \leq \text{tol}_\Lambda$ **then**
    Return x_{best}^- and F_k^{up}
  end if
  if $\Delta_k \leq \alpha\hat{\Delta}$ **then**
    Set $\hat{x}_k = x_{\text{best}}^-$ and $\hat{\Delta} = \Delta_k$.
    Possibly reduce J_k^j for all $j = 1, .., M$, but keep $\mathtt{a}(j) \in J_k^j$
  end if

Step 3: compute next iterate

  Set $F_k^{\text{lev}} = F_k^{\text{up}} - \alpha\Delta_k$. Run a quadratic solver on problem (7.6)
  if (7.6) is feasible **then**
    Get new iterate $x_{k+1} \in \mathbf{X}_k^{\text{d}}$. Update $F_{k+1}^{\text{low}} = F_k^{\text{low}}$
  else
    Set $F_k^{\text{low}} = F_k^{\text{lev}}$ and go to Step 2 ▷ update lower bound
  end if
  if $J_k^j \neq \emptyset$ for all $j = 1, .., M$ **then**
    $F_{k+1}^{\text{up}} = \min\left\{F_k^{\text{up}}, F_k^{\text{lev}} + \sum_{j=1}^M \left(\bar{\Lambda}^j \|x_{k+1} - x_{\mathtt{a}(j)}\| - \langle v_{\mathtt{a}(j)}^j, x_{k+1} - x_{\mathtt{a}(j)}\rangle\right)\right\}$
    if $F_{k+1}^{\text{up}} < F_k^{\text{up}}$ **then**
      set $x_{\text{best}}^- = x_{k+1}$
    end if
  else
    $F_{k+1}^{\text{up}} = F_k^{\text{up}}$
  end if

Step 4: send back information to the oracle

  Send x_{k+1} to machine i
  Set $\hat{x}_{k+1} = \hat{x}_k$
end for

Lemma 7.4. *Let Assumption 7.1 hold and suppose that (7.7) holds true for $i = 1, .., M$. At iteration k, whenever the master sub-problem (7.6) is feasible, with (x_{k+1}, r_{k+1}) its solution, one has*

$$F(x_{k+1}) \leq F_k^{\text{lev}} + \sum_{j=1}^M \left(\bar{\Lambda}^j \|x_{k+1} - x_{\mathtt{a}(j)}\| - \langle v_{\mathtt{a}(j)}^j, x_{k+1} - x_{\mathtt{a}(j)}\rangle\right). \tag{7.8}$$

Furthermore,

$$\alpha\Delta_k \leq F_k^{\text{up}} - F_{k+1}^{\text{up}} + 2\sum_{j=1}^M \bar{\Lambda}^j \|x_{k+1} - x_{\mathtt{a}(j)}\|. \tag{7.9}$$

Proof. Note first that

$$F(x_{k+1}) = \sum_{j=1}^M F^j(x_{k+1}) = \sum_{j=1}^M F^j(x_{\mathtt{a}(j)}) + \sum_{j=1}^M \left(F^j(x_{k+1}) - F^j(x_{\mathtt{a}(j)})\right).$$

This yields

$$F(x_{k+1}) \leq \sum_{j=1}^{M} \left(r_{k+1}^j - \langle v_{\mathrm{a}(j)}^j, x_{k+1} - x_{\mathrm{a}(j)} \rangle \right) + \sum_{j=1}^{M} \left(F^j(x_{k+1}) - F^j(x_{\mathrm{a}(j)}) \right)$$
$$\leq F_k^{\mathrm{lev}} + \sum_{j=1}^{M} \left(\bar{\Lambda}^j \|x_{k+1} - x_{\mathrm{a}(j)}\| - \langle v_{\mathrm{a}(j)}^j, x_{k+1} - x_{\mathrm{a}(j)} \rangle \right),$$

where the first inequality comes from the fact that (x_{k+1}, r_{k+1}) is feasible point for the master sub-problem (7.6). The second inequality uses that $\sum_{j=1}^{M} r_{k+1}^j \leq F_k^{\mathrm{lev}}$ as (x_{k+1}, r_{k+1}) is a feasible point of (7.6) and the Lipschitz assumption (7.7) for the functions.

To prove the second part of the result, we proceed as follows:

$$F_{k+1}^{\mathrm{up}} \leq F_k^{\mathrm{lev}} + \sum_{j=1}^{M} \left(\bar{\Lambda}^j \|x_{k+1} - x_{\mathrm{a}(j)}\| - \langle v_{\mathrm{a}(j)}^j, x_{k+1} - x_{\mathrm{a}(j)} \rangle \right)$$
$$\leq F_k^{\mathrm{up}} - \alpha \Delta_k + 2 \sum_{j=1}^{M} \bar{\Lambda}^j \|x_{k+1} - x_{\mathrm{a}(j)}\|,$$

where the second inequality is due to the definition of F_k^{lev} and due to the bound on the scalar product provided by the Lipschitz assumption (7.7). This concludes the proof. □

At each iteration, our asynchronous level algorithm (Algorithm 7.2) computes upper bounds of the functional values by using inequality (7.8). The rest of the algorithm corresponds essentially to the standard level algorithm (Algorithm 7.1) using the disaggregated model (7.6).[50] In particular, since the values F_k^{up} are provable upper bounds, we still have $F^\star \in [F_k^{\mathrm{low}}, F_k^{\mathrm{up}}]$ for all k and the stopping test $\Delta_k \leq \mathrm{tol}_\Delta$ is valid. The convergence analysis of the next section relies on proving that the sequence of gaps (Δ_k) goes to 0 when $\mathrm{tol}_\Delta = 0$.

[50] Note that Algorithm 7.2 needs non-infinite initial bounds F_1^{up} and F_1^{low}. These bounds can often be easily estimated from the data of the problem. Otherwise, we can call the M oracles at an initial point x_1 and wait for their first responses from which we can compute $F_1^{\mathrm{up}} = F(x_1) = \sum_{i=1}^{M} F^i(x_1)$ and F_1^{low} as the minimum of the linearization $F(x_1) + \langle v_1, x - x_1 \rangle$ over the compact set $\mathcal{X}$.

7.2.2 Convergence analysis

We denote by ℓ the number of *critical steps* that is the number of times line 14 is accessed, *i.e.* the number of times the gap significantly decreases. We note k_ℓ the corresponding iteration. We have by construction

$$\Delta_{k_{\ell+1}} \leq \alpha \Delta_{k_\ell} \leq \alpha^2 \Delta_{k_{\ell-1}} \leq \cdots \leq \alpha^\ell \Delta_1 \quad \forall \ell = 1, 2, \ldots \tag{7.10}$$

We call k_ℓ a *critical iteration*, and x_{k_ℓ} a *critical iterate*. We introduce the set of iterates between two consecutive critical iterates as illustrated by Fig. 7.2:

$$K_\ell := \{k_\ell + 1, \ldots, k_{\ell+1} - 1\}. \tag{7.11}$$

The proof of convergence of Algorithm 7.2 consists in showing the algorithm performs infinitely many critical iterations when $\mathrm{tol}_\Delta = 0$. We start with basic properties for iterations in K_ℓ, valid beyond Algorithm 7.2 for any level bundle method under mild assumptions. Observe that a critical iteration is due in exactly two situations: i) an update of F_k^{up}; or ii) an update of F_k^{low} from the infeasibility of the master sub-problem (7.6). The following lemma states that while i) can happen between two critical steps (*i.e.* in K_ℓ defined by (7.11)), the case ii) automatically triggers a critical step.

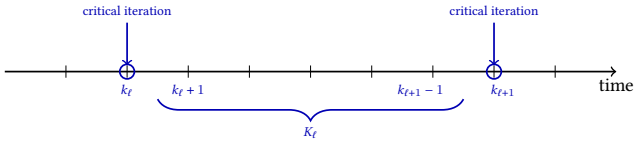


Figure 7.2: Illustration of the set K_ℓ used in convergence analysis

Lemma 7.5 (Between two consecutive critical iterates)**.** *Let Assumption 7.1 hold and consider a level bundle method (such as Algorithm 7.2) satisfying*

- F_k^{up} *is a non-increasing sequence of upper-bounds on* $F^\star$*;*
- F_k^{low} *is updated only when the master sub-problem is empty, and* F_k^{low} *is chosen as a lower bound greater or equal to* F_k^{lev}*;*
- F_k^{lev} *satisfies* $F_k^{\text{lev}} = \alpha F_k^{\text{low}} + (1-\alpha) F_k^{\text{up}}$*;*
- *all the linearizations are kept between two critical steps.*

Fix an arbitrary ℓ *and let* K_ℓ *be defined by (7.11). Then,* $(\mathbf{X}_k^{\text{d}})$ *is a nested non-increasing sequence of non-empty compact convex sets:* $\mathbf{X}_k^{\text{d}} \subset \mathbf{X}_{k-1}^{\text{d}}$ *for all* $k \in K_\ell$*. Furthermore, for all* $k \in K_\ell$*,*

(a) the master sub-problem (7.6) is feasible;

(b) the stability center and the lower bound are fixed: $\hat{x}_k = \hat{x}_{k_\ell}$ *and* $F_k^{\text{low}} = F_{k_\ell}^{\text{low}}$*;*

(c) the level parameter and the gap can only decrease: $F_k^{\text{lev}} \leq F_{k_\ell}^{\text{lev}}$ *and* $\Delta_k \leq \Delta_{k_\ell}$*.*

Proof. We start with proving (a), (b) and (c). Each $\mathbf{X}_k^{\text{d}}$ for $k \in K_\ell$ is non-empty as otherwise the master sub-problem (7.6) would be infeasible. Indeed, if (7.6) was infeasible at time k, F_k^{low} receives F_k^{lev} and therefore $F_k^{\text{low}} = F_k^{\text{up}} - \alpha\Delta_k \geq F_k^{\text{up}} - \alpha\Delta_{k_\ell}$ so $F_k^{\text{up}} - F_k^{\text{low}} \leq \alpha\Delta_{k_\ell}$, which contradicts the fact that $k \in K_\ell$ (*i.e.* is not a critical step). This proves (a).

For each $k \in K_\ell$, the stability center is fixed by construction and the lower bound is increased only when the master sub-problem (7.6) is found infeasible which is impossible for $k \in K_\ell$ (see above). This establishes (b).

The inequality on the level parameter comes directly as follows: $F_k^{\text{lev}} = \alpha F_k^{\text{low}} + (1-\alpha) F_k^{\text{up}} = \alpha F_{k_\ell}^{\text{low}} + (1-\alpha) F_k^{\text{up}} \leq \alpha F_{k_\ell}^{\text{low}} + (1-\alpha) F_{k_\ell}^{\text{up}} = F_{k_\ell}^{\text{lev}}$. Note finally that F_k^{up} is non-increasing and so is Δ_k. We thus have also (c).

Finally, each $\mathbf{X}_k^{\text{d}}$ is compact as $\mathcal{X}$ is compact, and it is also convex as $\mathcal{X}$ is convex and the disaggregate cutting-plane model is convex. The fact that $(\mathbf{X}_k^{\text{d}})$ is a nested non-increasing sequence is thus direct from (c) as the local cutting plane models only get richer with k (as the model is only reduced in critical steps which cannot happen in K_ℓ). □

We now provide a proof of convergence featuring elegant results from variational analysis (Rockafellar and Wets, 2009).

Theorem 7.6 (Convergence)**.** *Let Assumption 7.1 hold and suppose that (7.7) holds true for* $i = 1, .., M$*. Let* $\text{tol}_\Delta = 0$ *in Algorithm 7.2. Then, the sequence of gaps vanishes,* $\lim_k \Delta_k = 0$*, and the sequence of best iterates is a minimizing sequence for* $(\mathcal{P}_{\text{B-7}})$*,* $\lim_k F_k^{\text{up}} = F^\star$*. As a consequence, for a strictly positive tolerance* $\text{tol}_\Delta > 0$*, the algorithm terminates after finitely many steps with an approximate solution:* $F^\star \leq F(x_{\text{best}}^-) \leq F^\star + \text{tol}_\Delta$*.*

Proof. The convergence $\Delta_k \to 0$ is given by (7.10) as soon as the counter of critical steps ℓ increases indefinitely. Thus we just need to prove that, after finitely many steps, the algorithm performs a new critical iteration. For the sake of contradiction, suppose that only finitely many critical iterations are performed. Accordingly, let $\bar{\ell}$ be the total number of critical iterations and $k_{\bar{\ell}}$ be the index of the last critical iteration. Observe that $\hat{x}_k = \hat{x}$ is fixed and $\Delta_k \geq \Delta > 0$ for all $k > k_{\bar{\ell}}$. We have from Lemma 7.5, that $(\mathbb{X}_k^{\mathrm{d}})$ is a nested non-increasing sequence of non-empty compact convex sets for $k > k_{\bar{\ell}}$. Suppose that there is an infinite number of asynchronous iterations after the last critical iteration k_ℓ, then $(\mathbb{X}_k^{\mathrm{d}})$ converges to $\mathbb{X}^{\mathrm{d}}$ in the sense of the Painlevé-Kuratowski set convergence (Rockafellar and Wets, 2009, Chap. 4.B):

$$\lim_k \mathbb{X}_k^{\mathrm{d}} \;=\; \mathbb{X}^{\mathrm{d}} \;=\; \bigcap_k \mathrm{cl}\, \mathbb{X}_k^{\mathrm{d}}\,.$$

Now, Šmulian's theorem (Bernardes, 2012; Smulian, 1939) guarantees that the intersection $\mathbb{X}^{\mathrm{d}} = \cap_k \mathrm{cl}\, \mathbb{X}_k^{\mathrm{d}}$ is nonempty. Moreover, $\mathbb{X}^{\mathrm{d}}$ is by definition a convex compact set and, therefore, the projection of $\hat{x}$ onto $\mathbb{X}^{\mathrm{d}}$ is well defined and unique:

$$\mathrm{proj}_{\mathbb{X}^{\mathrm{d}}}(\hat{x}) = \mathrm{argmin}_{x \in \mathbb{X}^{\mathrm{d}}} \frac{1}{2}\|x - \hat{x}\|^2.$$

Then (Rockafellar and Wets, 2009, Prop. 4.9) implies that $x_{k+1} = \mathrm{argmin}_{x \in \mathbb{X}_k^{\mathrm{d}}} \frac{1}{2}\|x - \hat{x}\|^2 = \mathrm{proj}_{\mathbb{X}_k^{\mathrm{d}}}(\hat{x})$ converges to $\mathrm{proj}_{\mathbb{X}^{\mathrm{d}}}(\hat{x})$. Hence, (x_k) is a Cauchy sequence

$$\forall \varepsilon > 0\ \exists K \in \mathbb{N} \text{ such that } \forall k, k' \geq K \quad \Longrightarrow \quad \|x'_k - x_k\| \leq \varepsilon.$$

By taking $\varepsilon = \frac{\alpha}{4M \max_i \bar{\Lambda}^i}\Delta$ and $k \geq \min_{j=1,..,M} \mathtt{a}(j) \geq K$,[51] the inequality in (7.9) gives

$$\begin{aligned}\alpha\Delta \leq \alpha\Delta_k &\leq F_k^{\mathrm{up}} - F_{k+1}^{\mathrm{up}} + 2\sum_{j=1}^{M} \bar{\Lambda}^j \|x_{k+1} - x_{\mathtt{a}(j)}\| \\ &\leq F_k^{\mathrm{up}} - F_{k+1}^{\mathrm{up}} + 2\sum_{j=1}^{M} \bar{\Lambda}^j \frac{\alpha}{4M \max_i \bar{\Lambda}^i}\Delta \leq F_k^{\mathrm{up}} - F_{k+1}^{\mathrm{up}} + \frac{\alpha}{2}\Delta,\end{aligned}$$

showing that $F_k^{\mathrm{up}} - F_{k+1}^{\mathrm{up}} \geq \frac{\alpha}{2}\Delta > 0$ for all $k \geq \min_{j=1,..,M} \mathtt{a}(j) \geq K$. This is in contradiction with the fact that the sequence (F_k^{up}) is non-increasing and lower bounded, thus convergent. Hence, the index set K_ℓ is finite and ℓ grows indefinitely if $\mathrm{tol}_\Delta = 0$. Finally, the proof that (F_k^{up}) converges to the optimal value follows easily by noting that

$$F^\star \leq F_k^{\mathrm{up}} = F_k^{\mathrm{low}} + \Delta_k \leq F^\star + \Delta_k$$

from $F_k^{\mathrm{low}} \leq F^\star \leq F_k^{\mathrm{up}}$ and $\Delta_k = F_k^{\mathrm{up}} - F_k^{\mathrm{low}}$, which ends the proof. □

[51] As the oracles are assumed to respond in a finite time, the inequality $\min_{j=1,..,M} \mathtt{a}(j) \geq K$ is guaranteed to be satisfied for k is large enough.

7.3 Asynchronous level bundle by coordination

The level bundle algorithm of the previous section is fully asynchronous for solving problem ($\mathcal{P}_{\mathrm{B}-7}$). However, it relies on bounds on the Lipschitz constant of the functions F^i, which may be hard to have in practice. To compute upper-bounds F_k^{up}, we propose here an alternative strategy based on coordination of the machines to evaluate $F(x_k)$ only when necessary.

7.3.1 Coordinating to get upper-bounds

In our asynchronous setting, the oracles have no reason to be called at the same point except if the master decides to coordinate them. We propose a test that triggers a coordination on the points sent to the machines when the (proof of) convergence is in jeopardy (namely, when (7.4) does not hold). Thus, we introduce a coordination step (see line 32 in Algorithm 7.3): if the test is valid, this step consists in sending to all oracles the same iterate at the next iteration in which they are involved.[52] Finally, our coordination strategy does not generate idle times because the machines are not requested to abort their current jobs.

[52] To some extent, this can be seen as a variance reduction step resembling the scheme of SVRG (Johnson and Zhang, 2013), except that our coordination is only performed when needed and not after a fixed number of iterations.

This algorithm, given in Algorithm 7.3, corresponds to Algorithm 7.2 with more complex communications (Steps 1 and 4). Step 2 (optimality and sufficient decrease test) is unchanged. Finally, Step 3 (next iterate computation) relies on the same master sub-problem (7.6) in both algorithms, but here a coordination-triggering test replaces the upper-bound test (line 23 of Algorithm 7.2).

The coordination iterates are denoted by $\overline{x}_k$. Assuming that all oracles always eventually respond (after an unknown time), the coordination allows to compute the full value $F(\overline{x}_k)$ and a subgradient $v \in \partial F(\overline{x}_k)$ at the coordination iterate $\overline{x}_k$ (see lines 10 and 11, where `rr` ("*remaining to respond*") counts the number of oracles that have not responded yet to the coordination call). The functional value is used to update the upper bound F_k^{up}, as usual for level methods; the subgradient is used to update the bound $\bar{\Lambda}$ approximating the Lipschitz constant of F.

In the algorithm, the coordination is implemented by two vectors of booleans (to_coordinate and coordinating):

- The role to_coordinate$[i]$ is to indicate to machine i that its next computation has to be performed with the new coordination point $\overline{x}_{k+1}$; at that moment, to_coordinate$[i]$ is set to False and coordinating$[i]$ is set to True.
- The role coordinating$[i]$ is to indicate to the master that machine i is responding to a coordination step, which is used to update the upper bound (line 14).

We note that, as usual for level bundle methods, the sequence of the optimality gaps Δ_k is non-increasing by definition. We will further count with ℓ the number of times when Δ_k decreases *sufficiently*: more precisely, ℓ is increased whenever line 24 is accessed, and k_ℓ denotes the corresponding iteration index. As in the previous section, we call k_ℓ a critical iteration and consider K_ℓ the set of iterates between two consecutive critical iterates (recall (7.11) and Fig. 7.2).

7.3.2 Convergence analysis

We now turn to the convergence analysis of the asynchronous level bundle algorithm described in Algorithm 7.3. It features three types of iterates: the asynchronous iterates x_k, the coordination iterates $\overline{x}_k$, and the critical iterates x_{k_ℓ}, in addition to the stability centers $\hat{x}_k$.

As previously, we have by definition of critical iterates the chain of inequalities (7.10):

$$\Delta_{k_{\ell+1}} \leq \alpha \Delta_{k_\ell} \leq \ldots \leq \alpha^\ell \Delta_1 \quad \forall \ell = 1, 2, \ldots$$

and we rely on Lemma 7.5. The scheme of the convergence proof consists in showing that there exist infinitely many critical iterations. Note though that between two

Algorithm 7.3 Asynchronous level bundle method by coordination

Choose $x_1 \in \mathcal{X}$ and set $\overline{x}_1 = \hat{x}_1 = x_1$
Choose $\mathrm{tol}_\Delta \geq 0$, $\alpha \in (0,1)$, bounds $F_1^{\mathrm{up}} > F^\star + \mathrm{tol}_\Delta$ and $F_1^{\mathrm{low}} \leq F^\star$, and a constant $\bar{\Lambda} > 0$
Set $\hat{\Delta} = \infty$, $\mathtt{rr} = M$, ${J_0}^i = \emptyset$ for all $i = 1, .., M$, $\overline{F} = 0 \in \mathbb{R}$, and $\overline{v} = 0 \in \mathbb{R}^n$
Set to_coordinate$[i]$ = False and coordinating$[i]$ = True for all $i = 1, .., M$
Send x_1 to the M oracles
for $k = 1, 2, \ldots$ **do**

Step 1: receive information from oracle

  Receive from oracle i the oracle information on F^i at a previous iterate $x_{k'}$
  Store $(F^i(x_{k'}), v_{k'})$, and set $J_k^i = J_{k-1}^i \cup \{k'\}$
  if coordinating$[i]$ = True **then**
    $\mathtt{rr} \leftarrow \mathtt{rr} - 1$ and coordinating$[i] \leftarrow$ False
    Update $\overline{F} \leftarrow \overline{F} + F^i(x_{k'})$ and $\overline{v} \leftarrow \overline{v} + v_{k'}$
    if $\mathtt{rr} = 0$ **then** ▷ Full information at point $\overline{x}_k$
      if $\overline{F} < F_k^{\mathrm{up}}$ **then**
        Update $F_k^{\mathrm{up}} = \overline{F}$ and $x_{\mathrm{best}}^- = \overline{x}_k$ ▷ update upper bound
      end if
      Update $\bar{\Lambda} \leftarrow \max\{\bar{\Lambda}, \|\overline{v}\|\}$
    end if
  end if

Step 2: test optimality and sufficient decrease

  Set $\Delta_k = F_k^{\mathrm{up}} - F_k^{\mathrm{low}}$
  if $\Delta_k \leq \mathrm{tol}_\Delta$ **then**
    Return x_{best}^- and F_k^{up}
  end if
  if $\Delta_k \leq \alpha\hat{\Delta}$ **then** ▷ Critical Step
    Set $\hat{x}_k = x_{\mathrm{best}}^-$ and $\hat{\Delta} = \Delta_k$. Possibly reduce index sets J_k^j ($j = 1, .., M$)
  end if

Step 3: Compute next iterate

  Set $F_k^{\mathrm{lev}} = F_k^{\mathrm{up}} - \alpha\Delta_k$. Run a quadratic solver on problem (7.6)
  if (7.6) is feasible **then**
    Get new iterate $x_{k+1} \in \mathbb{X}_k^{\mathrm{d}}$. Update $F_{k+1}^{\mathrm{low}} = F_k^{\mathrm{low}}$, $F_{k+1}^{\mathrm{up}} = F_k^{\mathrm{up}}$
  else
    Set $F_k^{\mathrm{low}} = F_k^{\mathrm{lev}}$ and go to Step 2 ▷ update lower bound
  end if
  if $\mathtt{rr} = 0$ and $\|x_{k+1} - x_k\| < \alpha\frac{\Delta_k}{\bar{\Lambda}}$ **then**
    Set $\overline{x}_{k+1} = x_{k+1}$ and to_coordinate$[j]$ = True ($j = 1, .., M$) ▷ Coordination Step
    Reset $\mathtt{rr} = M$, $\overline{F} = 0$, $\overline{v} = 0$
  else
    Set $\overline{x}_{k+1} = \overline{x}_k$
  end if

Step 4: send back information to the oracle

  if to_coordinate$[i]$ = True **then**
    Send $\overline{x}_{k+1}$ to machine i.
    Set to_coordinate$[i]$ = False and coordinating$[i]$ = True
  else
    Send x_{k+1} to machine i
  end if
  Set $\hat{x}_{k+1} = \hat{x}_k$
end for

critical steps, we can have several coordination steps; the next result shows that two coordination iterates cannot be arbitrary close.

Lemma 7.7 (Between two coordination iterates). *Let Assumption 7.1 hold. For a given ℓ and two coordinate iterates $\overline{x}_s$ and $\overline{x}_t$ (with $\overline{x}_s \neq \overline{x}_t$) and $s < t \in K_\ell$, there holds*

$$\|\overline{x}_s - \overline{x}_t\| \geq \alpha \frac{\Delta_t}{\bar{\Lambda}}.$$

Proof. At the second coordinate iterate $\overline{x}_t \in \mathbf{X}^{\mathrm{d}}_{t-1}$, all the oracles have responded at least once and all of them have been evaluated at $\overline{x}_s$. Since the set of constraints of (7.6) keeps growing as k increase within K_ℓ, we have that $\overline{x}_t$ satisfies the M constraints generated by linearizations at $\overline{x}_s$. Summing these M linearizations and using that $\overline{x}_t \in \mathbf{X}^{\mathrm{d}}_{t-1}$ (see Eq. (7.5)) gives

$$F(\overline{x}_s) + \langle v_s, \overline{x}_t - \overline{x}_s \rangle = \sum_{i=1}^{M} \left(F^i(\overline{x}_s) + \langle v^i_s, \overline{x}_t - \overline{x}_s \rangle \right) \leq \sum_{i=1}^{M} F^i_{t-1}(\overline{x}_t) \leq F^{\mathrm{lev}}_{t-1}$$

This yields $-\|v_s\|\|\overline{x}_t - \overline{x}_s\| \leq F^{\mathrm{lev}}_{t-1} - F(\overline{x}_s)$ so that $\|\overline{x}_t - \overline{x}_s\| \geq (F(\overline{x}_s) - F^{\mathrm{lev}}_{t-1})/\|v_s\|$.

The value $F(\overline{x}_s)$ was fully computed before the next coordinate iterate, so before t. It is then used to update the upper bound, we have $F(\overline{x}_s) \geq F^{\mathrm{up}}_t$. This gives $F(\overline{x}_s) - F^{\mathrm{lev}}_{t-1} \geq F^{\mathrm{up}}_t - (F^{\mathrm{up}}_{t-1} - \alpha\Delta_{t-1}) \geq \alpha\Delta_{t-1} \geq \alpha\Delta_t$ where we used that (Δ_k) is non-increasing by construction. Finally, using the bound $\|v_s\| \leq \bar{\Lambda}$ as provided by the algorithm at line 16 completes the proof. □

Theorem 7.8 (Convergence). *Let Assumption 7.1 hold and assume that (7.7) holds. Let $\mathrm{tol}_\Delta = 0$ in Algorithm 7.3, then the sequence of gaps vanishes, $\lim_k \Delta_k = 0$, and the sequence of coordination iterates is a minimizing sequence for ($\mathcal{P}_{\mathrm{B-7}}$), $\lim_k F(\overline{x}_k) = F^\star$. For a strictly positive tolerance $\mathrm{tol}_\Delta > 0$, the algorithm terminates after finitely many steps with an approximate solution.*

Proof. The convergence $\Delta_k \to 0$ is given by (7.10), as soon as the counter ℓ increases indefinitely. Thus, we need to prove that there are infinitely many critical iterations. We obtain this by showing that, for any ℓ, the set K_ℓ is finite; for this, suppose that $\Delta_k > \Delta > 0$ for all $k \in K_\ell$. We proceed in two steps, showing that i) the number of coordination steps in K_ℓ is finite; and ii) the number of asynchronous iterations between two consecutive coordination steps is finite as well.

Part 1. Define $(\overline{x}_s)$ the sequence of coordination steps in K_ℓ. By Lemma 7.7, we obtain that for any $s < t$

$$\|\overline{x}_s - \overline{x}_t\| \geq \alpha \frac{\Delta_t}{\bar{\Lambda}} \geq \frac{\Delta}{\bar{\Lambda}}.$$

If there was an infinite number of coordination steps inside K_ℓ, the compactness of $\mathcal{X}$ would allow us to extract a converging subsequence, and this would contradict the above inequality. The number of coordination steps inside K_ℓ is thus finite.

Part 2. We turn to the number of asynchronous iterations between two consecutive coordination iterations, that is the number of iterations before the test of line 32 is active. This part is illustrated by the green arrows in Fig. 7.3.

Since all the oracles are responsive, there is a finite number of iterations between two updates of any oracle: as a consequence, at a given iteration k, there exists a T

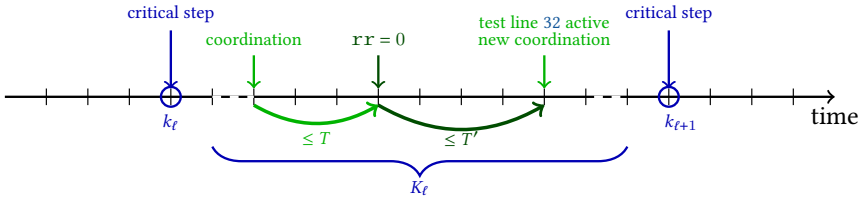


Figure 7.3: Notations for the proof

(the dependence on k is dropped for simplicity) such that all oracles will exchange at least twice in $[k, k+T]$; in other words, the first part of the test $\texttt{rr} = 0$ will be verified within a finite number T of iterations.

Now, let us show by contradiction that the second part of the test, $\|x_{k+1} - x_k\| < \alpha\Delta_k/\bar{\Lambda}$, will be verified after a finite number of iterations. As in the proof of Theorem 7.6, we have from Lemma 7.5 that $(\mathbb{X}_k^{\mathrm{d}})$ is a nested non-increasing sequence of non-empty compact convex sets for $k \in K_\ell$. If there was an infinite number of asynchronous iterations before the test is verified, the sequence $\mathbb{X}_k^{\mathrm{d}}$ would converge to a non-empty $\mathbb{X}^{\mathrm{d}}$ in the sense of the Painlevé-Kuratowski (see (Rockafellar and Wets, 2009, Chap. 4.B) and (Bernardes, 2012; Smulian, 1939)). As a consequence,

$$x_{k+1} = \operatorname{argmin}_{x \in \mathbb{X}_k^{\mathrm{d}}} \frac{1}{2}\|x - \hat{x}\|^2 \;\longrightarrow\; \operatorname{proj}_{\mathbb{X}^{\mathrm{d}}}(\hat{x}) = \operatorname{argmin}_{x \in \mathbb{X}^{\mathrm{d}}} \frac{1}{2}\|x - \hat{x}\|^2.$$

As (x_k) converges, for any $\varepsilon > 0$, there exists T' such that for any $k \geq T'$, $\|x_{k+1} - x_k\| \leq \varepsilon$. Taking $\varepsilon = \alpha\Delta/(2\bar{\Lambda})$ with $\Delta_k > \Delta > 0$, the second part of the test is verified after T' iterations which contradicts the infinite number of asynchronous iterations before the test is verified. Thus, there are at most $T + T'$ iterations between two coordination steps.

Combining i) and ii), we can then conclude that the algorithm performs only finitely many iterations between two consecutive critical iterations. This in turn shows that there are infinitely many critical iterations, and thus we get convergence using (7.10). Similarly, for a strictly positive tolerance, there are a finite number of critical steps, and thus a finite number of iterations. The result on the convergence of (F_k^{up}) follows from exactly the same arguments as in the end of the proof of Theorem 7.6. □

7.4 Extension to Inexact oracles

Many applications of optimization to real-life problems lead to objective functions that are assessed through noisy oracles, where only some approximations to the function and/or subgradient values are available; see *e.g.* the recent review (de Oliveira and Solodov, 2018). This is the typical case in Lagrangian relaxation of (possibly mixed-integer) optimization problems, in stochastic/robust programming, where the oracles perform some numerical procedure to evaluate functions and subgradients, such as solving optimization subproblems, multidimensional integrations, or simulations.

Level bundle methods are well-known to be sturdy to deal with such inexact oracles; see (de Oliveira and Sagastizábal, 2014; de Oliveira and Solodov, 2018; van Ackooij and de Oliveira, 2014). In particular, when the feasible set $\mathcal{X}$ is compact no special treatment is necessary to handle inexactness in standard level methods (de Oliveira and Solodov, 2018, Sec. 3). This is also the case for our asynchronous level bundle variants, more

precisely: i) the asynchronous algorithms converge to inexact solutions when used with oracles with bounded error (Section 7.4.1), and ii) with a slight modification of the upper bounds, they can converge to exact solutions when used with lower oracles with vanishing error (Section 7.4.2). For concision, we omit the proofs related to this section. They can be found in (Iutzeler et al., 2020, Sec. 5).

7.4.1 Inexact oracles

We assume to have an approximate oracle for F^i delivering, for each given $x \in \mathcal{X}$, an inexact linearization on F, namely $(F^i_x, v^i_x) \in \mathbb{R} \times \mathbb{R}^n$ such that

$$\left\{ \begin{array}{l} F^i_x = F^i(x) - \eta^{F,i}_x \\ v^i_x \in \mathbb{R}^n \quad \text{such that} \quad F^i(\cdot) \geq F^i_x + \langle v^i_x, \cdot - x \rangle - \eta^{v,i}_x \\ \text{with} \quad \eta^{F,i}_x \leq \frac{\eta^F}{M} \quad \text{and} \quad \eta^{v,i}_x \leq \frac{\eta^v}{M} \quad \text{for all } x \in \mathbb{R}^n. \end{array} \right. \tag{7.12}$$

The subscripts F and v on the oracle errors make the distinction between *function value* and *subgradient* errors. Note that oracles can overestimate function values as $\eta^{F,i}_x$ can be negative. In fact, both the errors $\eta^{F,i}_x$ and $\eta^{v,i}_x$ can be negative but not simultaneously because they satisfy $\eta^{F,i}_x + \eta^{v,i}_x \geq 0$.[53] Global bounds $\eta^F, \eta^v \geq 0$ on the errors should exist but are possibly unknown. When $\eta^v = 0$, we have the so-called lower oracles returning lower linearizations: $F^i(x) - \eta^{F,i}_x = F^i_x \leqslant F^i(x)$ and $F^i(\cdot) \geqslant F^i_x + \langle v^i_x, \cdot - x \rangle$. The exact oracle corresponds to taking $\eta^v = \eta^F = 0$.

[53] By substituting $F^i_x = F^i(x) - \eta^{F,i}_x$ in the inequality $F^i(\cdot) \geq F^i_x + \langle v^i_x, \cdot - x \rangle - \eta^{v,i}_x$ and evaluating at x, we get that $F^i(x) \geq F^i(x) - \eta^{F,i}_x - \eta^{v,i}_x$. This shows that $\eta^{F,i}_x + \eta^{v,i}_x \geq 0$.

Our asynchronous algorithms do not need to be changed to handle inexactness: the information provided by these inexact oracles is used in the same way as done previously where the oracles were exact. Indeed, we can define the inexact disaggregated cutting-plane model as

$$\check{F}^i_k(x) := \max_{j \in J^i_k} \{F^i_{x_j} + \langle v^i_{x_j}, x - x_j \rangle\}$$

and the inexact level set $\mathbb{X}^{\mathrm{d}}_k$ as in (7.5) (but with the inexact model). We then have the easy following result showing that F^{low}_k is still relevant.

Lemma 7.9 (Inexact lower bound). *The update of F^{low}_k in Algorithms 7.2 and 7.3 guarantees that it is an inexact lower bound, in the sense that*

$$F^{\mathrm{low}}_k \;\leq\; F^\star + \eta^v \qquad \text{for all } k\,. \tag{7.13}$$

In particular, if the oracle is a lower oracle ($\eta^v = 0$) then the algorithms ensure that F^{low}_k is a valid lower bound in every iteration k.

Similarly, the next lemma shows that F^{up}_k is relevant, up the oracle error.

Lemma 7.10 (Inexact upper bound). *The definitions of F^{up}_k in Algorithms 7.2 and 7.3 guarantee that it is an inexact upper bound; more precisely, at iteration k*

$$F^{\mathrm{up}}_k \;\geq\; F(x^-_{\mathrm{best}}) - \eta^F \;\geq\; F^\star - \eta^F. \tag{7.14}$$

The previous two lemmas thus show that the inexact upper and lower bounds appearing in the asynchronous algorithms with inexact oracles satisfy

$$F^{\mathrm{low}}_k - \eta^v \;\leq\; F^\star \;\leq\; F^{\mathrm{up}}_k + \eta^F \qquad \text{for all } k.$$

We now formalize a theorem to conclude that, as in the standard case, inexactness can be readily handled by our asynchronous level methods.

Theorem 7.11. *The convergence results for Algorithms 7.2 and 7.3, namely Theorems 7.6 and 7.8, still hold up to the oracle error when inexact oracles (7.12) are used. More precisely, we obtain a* $\mathrm{tol}_\Delta + \eta^F + \eta^v$*-solution of* ($\mathcal{P}_{B-7}$)*:*

- *when* $\mathrm{tol}_\Delta = 0$*, we have* $\quad \lim_k F_k^{\mathrm{up}} \le F^\star + \eta^F + \eta^v$
- *when* $\mathrm{tol}_\Delta > 0$*, we have* $\quad F^\star \le F(x_{\mathrm{best}}^-) \le F^\star + \mathrm{tol}_\Delta + \eta^F + \eta^v$

Proof. The proofs are valid verbatim until the end about the convergence of (F_k^{up}). In the inexact case, we combine (7.13) and (7.14) to write

$$F^\star - \eta^F \le F(x_{\mathrm{best}}^-) - \eta^F \le F_k^{\mathrm{up}} = F_k^{\mathrm{low}} + \Delta_k \le F^\star + \eta^v + \Delta_k.$$

Passing to the limit, this ends the proof. □

As previously mentioned, no special treatment is necessary to handle inexactness in the proposed asynchronous level methods. The obtained solution is optimal within the precision $\mathrm{tol}_\Delta + \eta^F + \eta^v$, the given tolerance plus the (possibly unknown) oracle error bounds. If we target obtaining tol_Δ-solutions, more assumptions on the inexact oracles need to come into play, and minor changes in the algorithms must be made, as explained in the next section.

7.4.2 Lower oracles with on-demand/vanishing error

We consider further the case of (7.12) with $\eta^v = 0$ and controllable η^F, for which the asynchronous algorithms converge to an optimal solution if we slightly change the upper bound.

We assume here that the error bound η^F is known and controllable in the sense that the algorithm can decrease or increase η^F along the iterative process. Thus we consider the following special case of (7.12): given a trial point x and an error bound η^F as inputs, the oracle provides

$$\begin{cases} F_x^i = F^i(x) - \eta_x^{F,i} \\ v_x^i \in \mathbb{R}^n \quad \text{such that} \quad F^i(\cdot) \ge F_x^i + \langle v_x^i, \cdot - x \rangle \\ \text{with} \quad \eta_x^{F,i} \le \frac{\eta^F}{M} \quad \text{for all } x \in \mathbb{R}^n. \end{cases} \tag{7.15}$$

The fact that we control η^F allows us to incorporate the oracle error in the algorithm, which eventually will yield convergence to optimality. The fact that $\eta^v = 0$ gives that F_k^{low} is always a lower bound (recall Lemma 7.9).

At iteration k we index the error bound with k and we drive η_k^F below a fraction of the gap at the preceding decrease. More precisely, we consider the following control: there exists $\kappa \in (0,1)$ such that the M oracles satisfy (7.15) with

$$0 \le \eta_k^F \le \kappa \Delta_{k_\ell}, \quad \text{for all } k \in K_\ell \tag{7.16}$$

where k_ℓ corresponds to the last critical iteration (the iteration yielding enough decrease on line 12 of Algorithm 7.2 or on line 24 of Algorithm 7.3). This control on η_k^F is standard in methods with on-demand accuracy (de Oliveira and Sagastizábal, 2014).

For Algorithm 7.2, we obtain the obtain the following result.

Theorem 7.12. *Consider Algorithm 7.2 with inexact oracles (7.15). If the oracles error can be controlled by (7.16) with* $\kappa \in (0, \alpha^2/2)$*, and if the update of* F_{k+1}^{up} *in line 23 is*

replaced with

$$\min\left\{F_k^{\mathrm{up}}, \left(F_k^{\mathrm{lev}} + \sum_{j=1}^{M}\left(\bar{\Lambda}^j\|x_{k+1} - x_{\mathrm{a}(j)}\| - \langle v^j_{\mathrm{a}(j)}, x_{k+1} - x_{\mathrm{a}(j)}\rangle\right) + \eta_k^F\right)\right\},$$

then the convergence result of Theorem 7.6 still holds.

And for Algorithm 7.3, we obtain a similar result with a different modification.

Theorem 7.13. *Consider Algorithm 7.3 with inexact oracles (7.15). If the oracles error are controlled by (7.16) with $\kappa \in (0, \alpha^2)$, then the convergence result of Theorem 7.8 still holds when the update of the upper bound of line 11 is replaced with*

$$\overline{F} \leftarrow \overline{F} + F^i_{x_\ell} + \frac{\eta^F_{k'}}{M}.$$

Thus we show that the two asynchronous algorithms also share the well-known robustness of synchronous level bundle methods for dealing with inexact oracles with on-demand accuracy (de Oliveira and Sagastizábal, 2014; van Ackooij and de Oliveira, 2014).

7.5 Numerical experiments

In this section, we provide illustrative numerical experiments to illustrate the effectiveness and the potential interest of asynchronicity in bundle methods. A thorough numerical assessing of the interests and limits of the algorithms would deserve a whole study of itself to take into account the various biases from the inputs and the computed system (in particular, the variance of the solution times of the numerical subroutines and the communications between machines). Here we consider a basic implementation of the (distributed) algorithms, a trivial set-up and computing system, and a simple randomly-generated problem.

7.5.1 Experimental set-up

Problem We consider an instance of problem ($\mathcal{P}_{\mathrm{B}-7}$) where each function F^i is the optimal value of the following mixed-integer linear program (MILP): for $x \in \mathbb{R}^n$,

$$F^i(x) = \left\{\begin{array}{rl} \max & \pi^i\left(\langle c^i, p\rangle - \langle x, A^i p\rangle\right) \\ \text{s.t.} & \|p\|_\infty \le B,\ \ G^i p \le h^i \\ & p \in \mathbb{N}^{n_1} \times \mathbb{R}^{n_2} \end{array}\right. \tag{7.17}$$

where c^i, A^i, G^i, h^i are random vectors/matrices with suitable sizes (we denote by n_c the number of affine constraints of the problem *i.e.* the number of lines of G^i, kept constant among oracles). Such oracles appear when solving Lagrangian relaxations of difficult mixed-integer optimization problems. The oracle i solves, for given a point x, the above MILP to get an optimal point $p^\star$, which gives

$$F^i(x) = \pi^i(\langle c^i, p^\star\rangle - \langle x, A^i p^\star\rangle) \quad \text{and} \quad v^i = -\pi^i A^i p^\star \in \partial F^i(x).$$

Tested algorithms We compare four following algorithms; two synchronous and two asynchronous level bundle methods:

- S Synchronous level bundle algorithm (Algorithm 7.1)

- SD Synchronous Disaggregated algorithm (Algorithm 7.1 using (7.6))
- AU Asynchronous algorithm by Upper-bounds (Algorithm 7.2)
- AC Asynchronous algorithm by Coordination (Algorithm 7.3)

The four algorithms use the same initialization and global parameters. The only exception is the level parameter α: we use the simple value $\alpha = 0.5$ except for AU, where we use $\alpha = 0.9$. This higher value allows us to better compensate for rough Lipschitz constant upper-bounds and get better levels F_k^{lev}.

Computing setup The code was written in Python 2.7.6[54] and run on a laptop with an Intel Core i7-5600U and 8GB of RAM. Each machine is assigned to a thread. MPI is used as a communication framework (more precisely the mpi4py implementation of OpenMPI 1.6.5). The MILPs (7.17) of the oracles and the quadratic problems at the master are computed using Gurobi 8.0.0.

[54] My bad... It was 2018...

Notice that for the standard algorithm S, the quadratic problem (7.2) uses the total function oracle while the disaggregated quadratic problem (7.6) uses all the oracles separately. In terms of distributed programming, the first one can be performed by map-reduce (with a sum operation in the reduce) while the second needs a separate gathering of the oracle results.

Instance generation We consider $M = 8$ machines/oracles on a problem size $n = 20$. We generate moderately imbalanced oracles: six comparable oracles ($n_1 = 20$, $n_2 = 40$, $B = 5$, $n_c = 100$, and $\pi^i = 1$) and two slightly bigger ($n_1 = 50$, $n_2 = 100$, $B = 10$, $n_c = 100$, and $\pi^i = 0.1$). The matrices A^i, G^i are drawn independently with coefficients taken from the uniform distribution in $(-1, 1)$, c^i is taken from the normal distribution with variance 100. Moreover h^i is taken from the uniform distribution in $[0.1, 1.1)$, so that $p = 0$ is feasible for all problems (7.17) and $F_0^{\text{low}} = 0$ is a valid lower-bound on $F^\star$.

Experiments With the above set-up, we run preliminary experiments. We observe a high variance of the solution times of Gurobi and the communication between machines: this strongly impacts the time for an oracle to respond, the order of oracles responses for asynchronous algorithms, and therefore the behavior of the algorithms. Note also that while the asynchronous methods have the same parameters as the standard level bundle methods, the behavior of the algorithm (*e.g.* the coordination frequency) highly depends on the problem and computing system. A complete computational study would be quite challenging; we focus here only on showing that using asynchronous methods can save time.

Thus, we generate, as described above, one problem instance for which the two bigger oracles (oracles 1 and 2) are computationally more expensive in practice. We consider five runs of the algorithms: the figures reported in next tables are the average (and the standard deviation) of the obtained results. We compare the algorithms, first for a coarse precision target (in Section 7.5.2), then for a finer precision (in Section 7.5.3). Finally, we investigate the case of inexact oracles (in Section 7.5.4).

7.5.2 Experiments for coarse precision

In this part, we stop the algorithms as soon as $\Delta_k / F^\star < 10\%$ and we display in Table 7.1 the number of iterations, the total CPU time, and number of oracle calls to reach this criterion. These figures illustrates the interest of disaggregation and asynchronicity.

Indeed, we first see that there is a real difference in computing time between the the usual level algorithm and the disaggregated ones proposed in this paper: 249s for the standard level bundle S vs 122s for its disaggregated counterpart SD, and as low as 53s for the best asynchronous method. We also see that the two asynchronous algorithms converge quickly compared to the synchronous ones: for example, S converges in 10 (synchronous) iterations which corresponds to 80 oracle calls whereas AC needs 192 oracle calls but its computing time is 5 times lower. We thus observe that synchronous methods are more reliable (less variance) and asynchronous ones may be faster (as they have a better use of wall clock time) even though they may compute more (they make more oracles calls).

Algo	# iters	F^1	F^2	F^3	F^4	F^5	F^6	F^7	F^8	time
S (Alg. 7.1)	10 = 80	10 ±0	10 ±0	10 ±0	10 ±0	10 ±0	10 ±0	10 ±0	10 ±0	249s ± 4s
SD	8 = 64	8 ±0	8 ±0	8 ±0	8 ±0	8 ±0	8 ±0	8 ±0	8 ±0	122s ± 6s
AU (Alg. 7.2)	232 ±60	14 ±6	18 ±9	31 ±8	33 ±9	33 ±9	32 ±9	31 ±9	32 ±9	78s ± 39s
AC (Alg. 7.3)	192 ±50	4 ±0	19 ±7	28 ±8	27 ±6	29 ±6	28 ±7	28 ±7	29 ±7	53s ± 29s

Table 7.1: Comparison of the four algorithms (in terms of number of iterations, number of oracles calls, and total computing time) in the case of low precision. We report the average and the standard deviation of the five results.

The two asynchronous algorithms reach the precision more quickly thanks to the asynchronous bundle information used to improve their lower-bounds, as showed in Fig. 7.4. In this figure, we see that synchronicity provides tight upper bounds to S and SD (and frequently w.r.t. the number of oracle calls) but they need more time to get good lower bounds. The greater variety of cuts added to the disaggregated master subproblem (7.6) by asynchronous methods enable them to enjoy better lower bounds than their synchronous counterparts. To close the gap, the asynchronous methods show different behaviors on upper-bounds. Indeed, we notice that the upper-bounds used by AU are weak with respect to the empirical estimation of the associated Lipschitz constants observed from norms of computed subgradients in AC; see Fig. 7.5. Thus, the two asynchronous algorithms AU and AC reach the prescribed coarse precision faster than the synchronous ones, with roughly the same time. However, the loose upper-bounds in AU make it less competitive as the precision becomes finer.

7.5.3 Experiments with finer precision

For our second experiments, we stop the algorithms whenever $\Delta_k/F^\star < 1\%$. In order to precise the reach of our methods, we focus here on our flagship asynchronous algorithm with coordination AC and compare with the synchronous baseline S. We notably illustrate the impact of the proposed coordination strategy by investigating two variants of AC: when the test of line 32 is 'on' or 'off', 'off': corresponding to the case where a coordination is triggered as soon as the previous one has been completed. In the following table, we again display the average on 5 runs as well as the standard deviation.

We notice that the asynchronous algorithms achieve a clear speedup compared to the synchronous bundle. This can be explained intuitively by the fact that the first

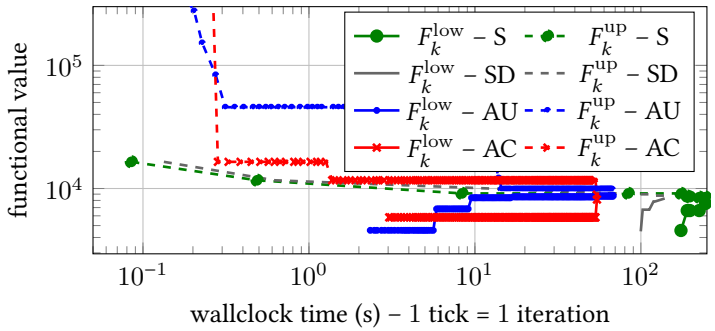


Figure 7.4: Evolution of F_k^{low} and F_k^{up} for on representative run of the algorithms. The two axis (functional values and iterations) are in log-scale.

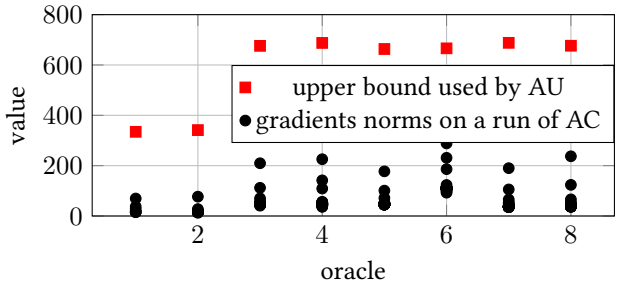


Figure 7.5: Estimation of the Lipschitz constants for the oracles: *a priori* upper bounds for AU vs. observed norms of computed subgradients.

Algo	# iters	F^1	F^2	F^3	F^4	F^5	F^6	F^7	F^8	time
S (Alg. 7.1)	20 = 160	20 ±0	20 ±0	20 ±0	20 ±0	20 ±0	20 ±0	20 ±0	20 ±0	390s ± 7s
AC (Alg. 7.3)	285 ±57	6 ±1	32 ±10	41 ±9	40 ±7	42 ±7	41 ±8	41 ±8	42 ±8	116s ± 37s
AC test off	294 ±91	5 ±1	32 ±14	42 ±12	42 ±13	45 ±13	41 ±13	43 ±13	44 ±13	128s ± 63s

Table 7.2: Comparison of the asynchronous algorithm AC with the baseline S (in terms of number of iterations, number of oracles calls, and total computing time) in the case of high precision. We illustrate the impact of the coordination step of line 32 by looking at AC when the test is turned 'off'. We report the average and standard deviation on 5 runs.

two oracles are more time consuming than the other (as their associated subproblem is harder to solve) while they do not contribute proportionally more in the global model. The asynchronous algorithms thus achieve the sought precision after only 5 or 6 calls from oracle 1 and around 40 for the others while the synchronous one has to get 20 global calls.

In Fig. 7.6, we plot the values of F_k^{low} and F_k^{up} computed along the runs versus the number of oracle calls. We notice that while the synchronous method improves iteration by iteration (there are 8 calls per iteration), the asynchronous algorithm

improves more scarcely but with more significant decreases, often made after receiving a *hard* oracle (from #1 or #2). Due to the difference in terms of computational cost between the workers, one has to keep in mind that the wallclock time cost of a certain number of oracle calls is smaller in the asynchronous setup, which allows for faster convergence.

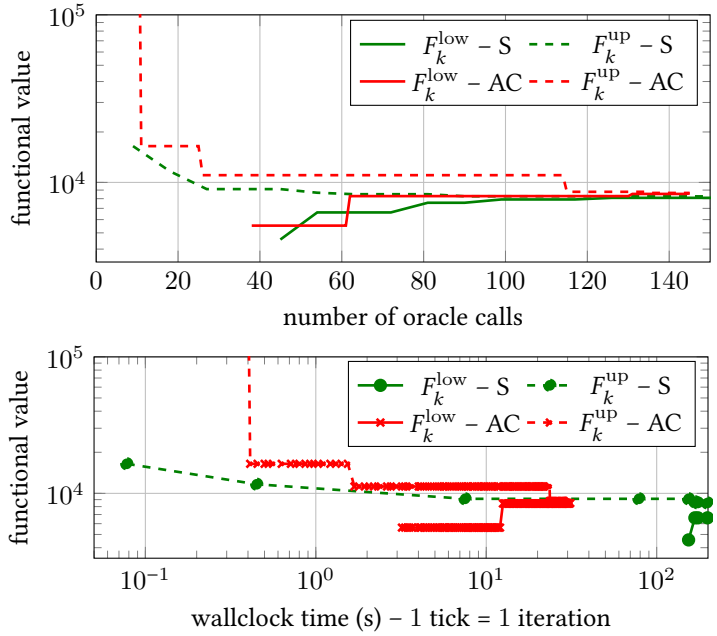


Figure 7.6: Evolution of F_k^{low} and F_k^{up} for S and AC for a representative run.

7.5.4 Experiments with inexact oracles

In this section, we compare our asynchronous algorithms AU and AC with their on-demand accuracy counterparts from Section 7.4.

We control the accuracy of the oracles by sending to the workers a target precision along with the trial point. More precisely, at iteration k, we send $\kappa\Delta_k/F_k^{\text{lev}}$ as a target (relative) precision, used by the worker as the precision-control parameter MIPGap of Gurobi. The parameter κ was chosen equal to 0.001 which, given the functional values, corresponds to a relative precision lowering from 10 in the first iterations to 10^{-3} in the final steps; compared to a fixed precision of 10^{-9} for AU and AC. We stop the algorithms whenever $\Delta_k/F^\star < 3\%$, corresponding to an intermediate precision compared to the two previous sections. The rest of the setup is exactly the same as in the previous section.

In Table 7.3, we display the average on 5 runs as well as the standard deviation. The use of inexact oracles seems to speed-up the convergence of AU and AC both in terms of wall-clock time and number of oracle calls (with a more even number of calls across the oracles); this could be due to the fact that larger gaps in the first iterations would make an exact oracle too expensive for the potential gain in the master bundle.

Algo	# iters	F^1	F^1	F^3	F^4	F^5	F^6	F^7	F^8	time
AU (Alg. 7.2)	308 ±104	14 ±6	24 ±11	32 ±15	40 ±14	51 ±15	47 ±15	49 ±15	50 ±15	132s ± 76s
AU **on-demand**	320 ±73	40 ±10	40 ±9	38 ±7	40 ±9	41 ±9	40 ±9	41 ±10	41 ±9	129s ±56s
AC (Alg. 7.3)	219 ±46	4 ±1	18 ±6	25 ±8	30 ±6	36 ±7	35 ±7	35 ±7	37 ±8	67s ± 26s
AC **on-demand**	97 ±9	12 ±1	12 ±1	11 ±2	12 ±1	13 ±1	12 ±1	12 ±1	13 ±1	14s ± 2s

Table 7.3: Comparison of the two asynchronous algorithms and their counterparts with on-demand accuracy (in terms of number of iterations, number of oracles calls, and total computing time). We report the average and the standard deviation on 5 runs.

7.6 Concluding remarks

We focused in this chapter on the main ideas that enable to produce efficient (level) bundle methods from several asynchronous first-order oracles. To our knowledge, these were the first bundle methods that can address asynchrony efficiently in the sense that the oracles are tirelessly called upon new points with no idle times nor any knowledge about the computing system.

We payed a special attention to presenting our study of asynchrony in a comprehensive way, by developing i) a disaggregated level bundle method, ii) a simple fully asynchronous algorithm (requiring additional assumptions), and iii) an advanced flexible asynchronous algorithm with automatic coordination.

An important finding is that perhaps the central issue in asynchronous bundle is the generation of valid upper bounds. To do so, we showed that a direct approach by using a Lipschitz bound on the objective is possible. Nevertheless, this techniques suffers from the possibly high variance of the points at which the oracle may respond which makes the method rather sensitive. To overcome this, we got back to the proof and identified when a the knowledge of the full function value was useful to get a precise upper-bound. This lead to an algorithm which automatically decides (by a simple test) when to coordinate the oracles on one point. This does not break our asynchrony paradigm since the workers are simply given a common for their next exchange with the master (their current information still being used to enrich the master sub-problem). This technique practically works better than its synchronous counter part but also with other strategies that coordinate more often.

Pushing further the reasoning developed in this chapter, it is possible to foresee several possible improvements: i) dealing with unbounded constraint sets, ii) having limited memory (bounded storage of information), and iii) coping with mixed-integer feasible sets and inexact solution of the resulting master program. Another direction would be to derive a complexity analysis of our methods. However, such a result would probably be overly pessimistic and not in line with the targeted computing situations.

8 Asynchronous Distributed Optimization

César – *Compression de motocycle (1970)*

We focus on the asynchronous distributed minimization of objectives that can be written as a sum of smooth functions, local to each worker, and a non-smooth function. We first develop an efficient extension of the proximal gradient to handle asynchronous oracles. Then, building on the structure analysis carried in Part A, we provide a technique to reduce the communication cost based on the identification of the optimal sparsity pattern.

This chapter is based on the following publications:

- K. Mishchenko, F. Iutzeler, and J. Malick : A Distributed Flexible Delay-tolerant Proximal Gradient Algorithm , SIAM Journal on Optimization, vol. 30, no. 1, pp. 933-959, 2020.
- D. Grishchenko, F. Iutzeler, J. Malick, and M.-R. Amini: Distributed Learning with Sparse Communications by Identification , SIAM Journal on Mathematics of Data Science, vol. 3, no. 2, pp. 715-735, 2021.
- K. Mishchenko, F. Iutzeler, J. Malick, M.-R. Amini: A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning, 35-th International Conference on Machine Learning (ICML), PMLR 80:3584-3592, Stockholm (Sweden), July 2018.

In this chapter, we will consider a particular case of ($\mathcal{P}_{\mathrm{B}}$) where the functions F^i at the workers are of the form

$$F^i(x) \propto f^i(x) + g(x)$$

where f^i is a local smooth function and g is a global nonsmooth function. This setting typically arises in machine and signal processing where f^i may represent a local loss and g a non-smooth regularizer that imposes some structure on optimal solutions, as discussed in Part A.

In this chapter, we consider that the M workers can compute:

- the gradient of their local function ∇f^i;
- the proximity operator of the common nonsmooth function prox_g;

while a coordinator handles the communication between the agents. We focus in particular on the setup where (i) the workers' functions differ in their values and the computational complexity of their local oracles (*e.g.* due to non-i.i.d. unbalanced local datasets in a learning scenario); (ii) the communications between workers and the coordinator are time-consuming (*e.g.* due to scarce availability or slow communications). This implies that we need to pay a special attention to the delays of workers' updates.

In this distributed setting, we provide an asynchronous proximal gradient algorithm and the associated analysis that adapts to local functions' parameters and can handle any kind of delays. In order to subsume delays, we develop an epoch-based mathematical analysis, encompassing computation times and communication delays, to refocus the theory on algorithmics.

8.1 Distributed Averaging of (Repeated) Proximal Gradient steps

Formally, we will consider functions F^i of the form

$$F^i(x) = \frac{1}{M}\left(f^i(x) + g(x)\right)$$

which corresponds to a global objective with an additive smooth structure plus a nonsmooth part

$$\min_{x\in\mathbb{R}^n} F(x) := \sum_{i=1}^{M} F^i(x) = \frac{1}{M}\sum_{i=1}^{M} f^i(x) + g(x). \qquad (\mathcal{P}_{\mathrm{B-8}})$$

We make the following blanket assumption.

Assumption 8.1. For each $i = 1, .., M$, the function $f^i : \mathbb{R}^n \to \mathbb{R}$ is μ^i-strongly convex and L^i-smooth. The function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is convex, proper, and lower semi-continuous.

This corresponds to the general case where the agents' (smooth) functions may differ across the agents, for instance if they have different types of objectives or different number of examples. In this case, it is important that they can choose different stepsizes as we will do in this section.

When the agents' (smooth) functions are similar and only the number of examples change, a single stepsize strategy can be alternatively adopted by directly rescaling the functions and considering a weighted average of these new objectives. This case is considered in Section 8.3.2.

In this section, we present the proposed DAve-RPG algorithm, where DAve stands for the global communication scheme based on distributed averaging of iterates, and RPG stands for the local optimization scheme, based on repeated proximal-gradient steps. We start by presenting the generic coordinator worker setting and associated notations.

8.1.1 Asynchronous Coordinator–Worker Framework

In order to analyze our methods in the asynchronous setup common to this part, we adopt the following notations. As before, we call iteration/time k, the moment of the k-th exchange between a worker and the coordinator, or, equivalently, the k-th time the coordinator has updated its coordinator variable. However, we will now handle the delays more explicitly: we denote by d_k^i the delay for i at time k, *i.e.* the number of coordinator updates since worker i's last exchange with the coordinator. More precisely, at time k, the updating worker $i = i_k$ suffers no delay (in terms of update in the coordinator variable), *i.e.* $d_k^i = 0$, while the delays of the other workers are incremented ($d_k^j = d_{k-1}^j + 1$ for all $j \neq i_k$). In addition, we denote by D_k^i the relative delay from the penultimate update, mathematically defined as $D_k^j = d_k^j + d^j_{k-d_k^j-1} + 1$ for worker j and time k; see Fig. 8.1.

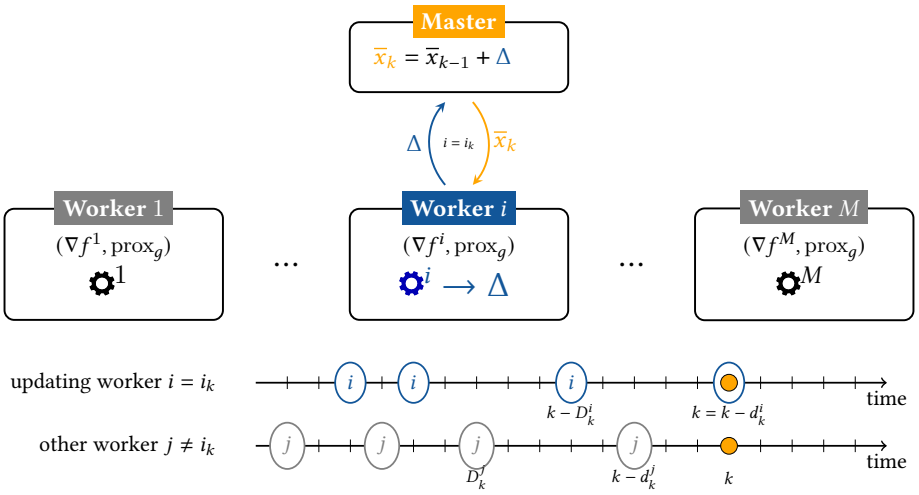


Figure 8.1: Asynchronous distributed setting and delays notations at iteration k.

As mentioned in the introduction to this part, we do *not* assume that the delays are uniformly bounded.

8.1.2 DAve Communication scheme

Our communication scheme is based on maintaining at the coordinator the weighted average of the most updated parameters of the workers. At time k, worker $i = i_k$ finishes the computation of a new *local parameter* x_k^i and the corresponding *adjustment* Δ corresponding to the weighted difference between its new and former local parameter. As soon as the computation is finished, this adjustment is sent to the coordinator node which, in turn, adds it to its *coordinator parameter* $\overline{x}_k$. The coordinator then

immediately sends back this parameter to worker i, which can begin a new computation step. During the updates, the coordinator variable is "locked" (see *e.g.* the description of (Peng et al., 2016)), guaranteeing read/write consistency. This way, the workers can compute their updates without interrupting or waiting for each other.

Mathematically, at each time k, one has

$$\overline{x}_k = \overline{x}_{k-1} + \Delta \text{ with } \Delta = \pi^i(x_k^i - x^i_{k-D_k^i}) \text{ for } i = i_k$$
$$\text{thus,} \quad \overline{x}_k = \sum_{i=1}^M \pi^i x^i_{k-d_k^i} = \sum_{i=1}^M \pi^i \text{⚙}^i(\overline{x}_{k-D_k^i}) \tag{8.1}$$

where ⚙^{i} represents the computation of worker i (see Fig. 8.1) and the $(\pi^i)_{i=1,..,M}$ are the weights of the workers contributions. These weights are positive real numbers such that $\sum_{i=1}^M \pi^i = 1$ and are kept fixed over time.[55]

[55] As we will see next, their values are derived from the optimality conditions of ($\mathcal{P}_{B-8}$) and the workers computation. In this chapter, the agents will perform (proximal) gradient steps (⚙^{i} = proximal gradient step on $f^i + g$) which leads to the weights given by (8.3).

We see in Eq. (8.1) that $\overline{x}_k$ depends on local parameters $(x^i_{k-d_k^i})_i$, which themselves were computed using (once more delayed) global parameters $(\overline{x}_{k-D_k^i})_i$. We note that this idea of averaging iterates has also been used in the different context of variance reduction in incremental methods (Defazio et al., 2014b; Mokhtari et al., 2018). This means that the contribution of each worker in the coordinator variable stays fixed over time even if one worker updates much more frequently than the others. Though this simple idea might be counterproductive in other contexts, it allows the algorithm to cope with heterogeneity in the computing system such as data distribution and agents delays. Roughly speaking, in standard approaches, if an agent has very outdated information, the output of its computation can lead to a counter-productive change, generating instability in the algorithm; keeping a fixed average of the contributions offers a counterbalance to such drastic updates. This phenomenon is illustrated in the case where the agents computations are gradients steps in Section 8.1.4, notably through Figs. 8.2 and 8.3.

8.1.3 Optimization scheme: Repeated Proximal Gradient RPG

As the problem features a smooth and a non-smooth part, it is natural that the workers use proximal gradient steps. Furthermore, we allow the repetition of local proximal gradient steps before exchanging with the coordinator, for higher flexibility in the computing time between two exchanges. We present our RPG scheme in 3 stages, explaining the three letters of the name. For more readability, we consider a generic worker i and time k when $i = i_k$ is the exchanging worker (as represented in Fig. 8.1).

♦ **G.** If $g \equiv 0$, then each worker may perform a simple gradient step on the last coordinator parameter received $\overline{x}_{k-D_k^i}$:

$$x_k^i \leftarrow \overline{x}_{k-D_k^i} - \gamma^i \nabla f^i(\overline{x}_{k-D_k^i}), \qquad \Delta \leftarrow \pi^i \left(x_k^i - x^i_{k-D_k^i}\right) \tag{8.2}$$

where γ^i is the *local stepsize* at worker i (related only to function f^i) and

$$\pi^i := \frac{\frac{1}{\gamma^i}}{\sum_{j=1}^M \frac{1}{\gamma^j}} \tag{8.3}$$

is the *proportion* of worker i's contribution, necessary to converge to the correct point.

♦ **PG.** For a general non-smooth convex function g, one can extend (8.2) using a proximity operator. However, contrary to direct intuition, the proximity operator has to be computed first, leading to a temporary variable u, on which is taken the gradient step before exchanging:[56]

$$u \leftarrow \mathrm{prox}_{\gamma g}(\overline{x}_{k-D_k^i}), \quad x_k^i \leftarrow u - \gamma^i \nabla f^i(u), \quad \Delta \leftarrow \pi^i \left(x_k^i - x_{k-D_k^i}^i\right) \tag{8.4}$$

with γ being the *coordinator stepsize* appearing in *all* proximity operators:

$$\gamma := \frac{M}{\sum_{i=1}^M \frac{1}{\gamma^i}} \tag{8.5}$$

equal to the harmonic average of the local stepsizes. Note that our algorithm allows for different local stepsizes, which simplifies parameters tuning as it can be done locally. Then, the proximity operators have to be taken with a separate coordinator stepsize.

♦ **RPG.** Once all computations of iteration (8.4) are done, the worker could send the adjustment Δ to the coordinator and get $\overline{x}_k$ in response. However, the difference between the latest $\overline{x}_k$ and $\overline{x}_{k-D_k^i}$ may be small, so the worker would only gain little information from a new exchange. Thus, instead of communicating right away, we suggest to perform additional proximal gradient updates by taking as the starting point $\overline{x}_{k-D_k^i} + \Delta$. The motivation behind this repetition is to lower the burden of communications and to focus on computing good updates. We will prove later that there is no restriction on the number of repetitions (called p in the algorithm), as any value can be chosen and it can vary freely both across machines and over time.

Our full method DAve-RPG is displayed as Algorithm 8.1.[57]

[56] Actually, this is simply due to the fact that the proximity operator of $(x^1, .., x^M) \mapsto \iota_{x^1=..=x^M}(x^1, .., x^M) + \sum_{=i}^M{}_1 g(x^i)$ is $\mathrm{prox}_g(\overline{x})$ where $\overline{x}$ is the average of the (x^i). Hence, the proximity operator of g has to be computed just after the averaging step.

[57] To prepare for following developments, we explicitly mention an abstract stopping test C in the algorithm. Moreover, one can notice that only the iterates at the coordinator are numbered. This is to reflect that the master updates are the actions that trigger a new iteration, the notion of iteration has no real meaning on the worker side.

Algorithm 8.1 DAve-RPG for ($\mathcal{P}_{B-8}$)

Coordinator:

Initialize $\overline{x} = \overline{x}_0$, $k = 0$
while test C not verified **do**
  Receive adjustment Δ_k from worker i_k
  $\overline{x}_k = \overline{x}_{k-1} + \Delta_k$
  Send $\overline{x}_k$ to the worker in return
  $k \leftarrow k + 1$
end while
Interrupt all workers
Output $x_k = \mathrm{prox}_{\gamma g}(\overline{x}_k)$

Worker i:

Initialize $x = x^i = \overline{x}$
while not interrupted by coordinator **do**
  Receive $\overline{x}$ from the coordinator
  Select a number of repetitions p
  $\Delta \leftarrow 0$
  for $q = 1, .., p$ **do**
    $u \leftarrow \mathrm{prox}_{\gamma g}(\overline{x} + \Delta)$
    $y \leftarrow u - \gamma^i \nabla f^i(u)$
    $\Delta \leftarrow \Delta + \pi^i (y - x)$
    $x \leftarrow y$
  end for
  Send adjustment Δ to the coordinator
end while

8.1.4 Comparison between our averaging and other incremental method

Our algorithm performs a distributed minimization of the composite problem ($\mathcal{P}_{B-8}$) by aggregating the agents contributions. It is closely related to the proximal incremental aggregated gradient (PIAG) method (Aytekin et al., 2016; Vanli et al., 2018). We can compare the update of PIAG with the one of $x_k = \text{prox}_{\gamma g}(\overline{x}_k)$ for DAve-PG (with one repetition, $p = 1$).[58]

DAve-PG	PIAG
$x_k = \text{prox}_{\gamma g}\left(\sum_{i=1}^M \pi^i x_{k-D_k^i} - \sum_{i=1}^M \pi^i \gamma^i \nabla f^i(x_{k-D_k^i})\right)$ $= \text{prox}_{\gamma g}\left(\sum_{i=1}^M \pi^i x_{k-D_k^i} - \gamma \frac{1}{M} \sum_{i=1}^M \nabla f^i(x_{k-D_k^i})\right)$	$x_k = \text{prox}_{\gamma g}\left(x_{k-1} - \gamma \frac{1}{M} \sum_{i=1}^M \nabla f^i(x_{k-D_k^i})\right)$

These two algorithms are separated by a major difference: PIAG performs an aggregated delayed gradient descent from the most recent main variable x_{k-1} and uses all gradients regardless of corresponding delays. Clearly, if one gradient has not been updated for long time, this update rule may be harmful. On the other hand, DAve-(R)PG performs a similar aggregated delayed gradient descent (with more adaptive local stepsizes) but from the averaged main point $\tilde{x}_{k-1} := \sum_{i=1}^M \pi^i x_{k-D_k^i}$. This more conservative update prevents instabilities in the case where some worker is silent for too long, and, thus, is more robust. See Fig. 8.2 for a geometrical illustration.

This difference is intuitively the same as the one between *e.g.* SAG (Schmidt et al., 2017) and MISO (Mairal, 2015) in the context of variance-reduced stochastic gradients. In these methods, one agent is sampled, returns its gradient, and the sum of the last computed gradients is used as a direction. The main difference between SAG and MISO is that this direction is respectively applied to the last (as PIAG), or the average point (as DAve-PG).

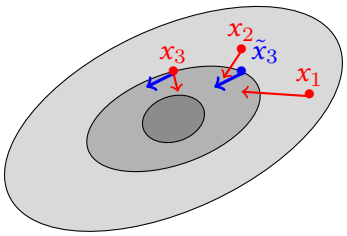


Figure 8.2: Let the gray ellipses be the level-sets of a smooth function. In red are represented three iterates $(x_k)_{k=1,2,3}$ and their associated descent directions (taken as the opposite of the gradients computed at these points). The blue dots represent the averaged point $\tilde{x}_3 = (x_1 + x_2 + x_3)/3$, while the blue vectors both represent the average of the associated descent directions. We notice that in that situation, descending along the averaged gradient is much more interesting from the averaged point $\tilde{x}_3$ than from the last point x_3.

In terms of theoretical results, this conservative approach allows us to get stronger convergence results and better rates as derived in the next section:

- the stepsize of PIAG, and thus its rate, depends heavily on the maximal delays whereas our stepsize does not depend on any form of delays;
- PIAG's stepsize is global and thus cannot adapt to each of the workers local functions, while we use locally adapted stepsizes;

[58] For the coordinator, the iteration k reads $x_k = \text{prox}_{\gamma g}(\overline{x}_k)$ where $\overline{x}_k$ is the average of the last update of each worker: $\overline{x}_k = \sum_{i=1}^M \pi^i x_k^i$ (see Eq. (8.1)). For each worker i, x_k^i is the result of the last gradient step performed by this worker on its local function: $x_k^i = x_{k-D_k^i} - \gamma^i \nabla f^i(x_{k-D_k^i})$ (see Eq. (8.2)). Putting it all together, we get $x_k = \text{prox}_{\gamma g}(\sum_{i=1}^M \pi^i x_{k-D_k^i} - \sum_{i=1}^M \pi^i \gamma^i \nabla f^i(x_{k-D_k^i}))$. Finally, this expression can be simplified by noticing that $\pi^i \gamma^i = \gamma / M$ (see Eqs. (8.3) and (8.5)).

- no version of PIAG exists with *multiple* proximal gradient steps before exchanging with the coordinator.

In terms of performance, before more thorough comparisons, Fig. 8.3 gives an illustration of the benefits of the averaged approach in terms of iterates behavior. In this plot, we consider two runs of DAve-RPG and PIAG applied to a two dimensional problem where one of the 5 functions/workers takes 10 times as much time to compute its update as the other workers and consequently produces more delayed updates. The objective used is a sum of 5 quadratics centered around different points and the initial point is (-20, -20) in all cases. Although the stepsize used for PIAG was 10 times smaller (due to its dependence to the delays), the iterates produced by PIAG show chaotic deviations from the optimal point while DAve-RPG steadily converges to the optimum.

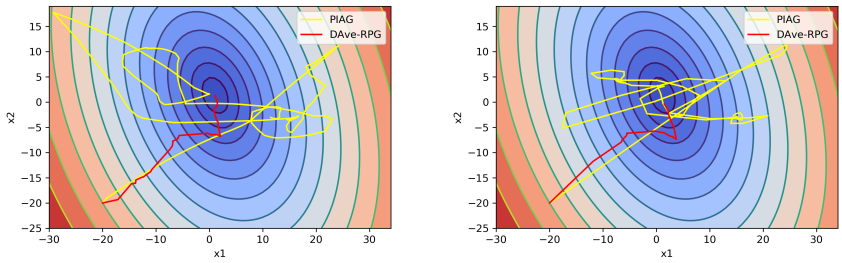


Figure 8.3: Two runs of a two dimensional example with $M = 5$ and one worker suffering long delays.

8.2 Analysis

8.2.1 Revisiting the clock

To the best of our knowledge, all papers on asynchronous distributed methods (except (Hannah and Yin, 2016, 2017; Sun et al., 2017)) assume that delays are uniformly upper bounded by a constant. Moreover, the maximum stepsize is usually highly dependent on this upper bound. In the upcoming results, we show that our algorithm DAve-RPG converges without assuming bounded delays and with the stepsizes depending only on local smoothness and convexity of the functions.

The forthcoming results are based on the careful definition of an *epoch sequence* along which we investigate the improvement of our algorithm (rather than looking at the improvement per iteration).

We define our *epochs sequence* $(k_\ell)_\ell$ by setting $k_0 = 0$ and the recursion:

$$\begin{aligned} k_{\ell+1} &= \min\{k : \text{ each machine made at least 2 updates on the interval } [k_\ell, k]\} \\ &= \min\{k : k - D_k^i \geq k_\ell \text{ for all } i = 1, .., M\}. \end{aligned}$$

In words, k_ℓ is the first moment when all workers have updated twice since $k_{\ell-1}$. This is illustrated by Fig. 8.4. Thus, k_ℓ is the first moment when $\overline{x}_k$ no longer depends directly on information from moments before $k_{\ell-1}$. Indeed, we have $\overline{x}_k = \sum_i \pi^i x^i_{k-d_k^i}$ and $x^i_{k-d_k^i}$ was computed using $\overline{x}_{k-D_k^i}$.

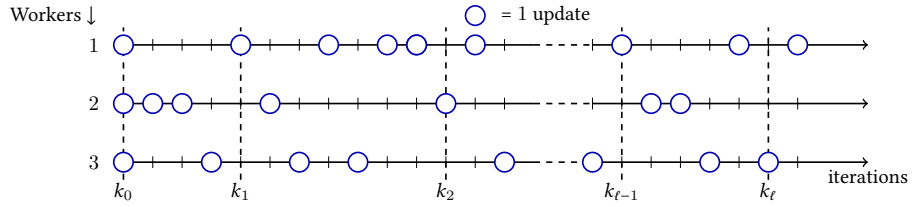


Figure 8.4: Illustration of the epoch sequence for $M = 3$ workers. Each circle corresponds to one update, *i.e.* one iteration.

Note that we always have $k_\ell \geq 2M - 1$. Furthermore, in the degenerate case when $M = 1$, the epoch sequence corresponds to the time sequence: we have $k_\ell = \ell$, because on the interval $[\ell, \ell + 1]$ there are exactly two updates of the only worker. In addition, we will assume that the number of epochs goes to infinity, *i.e.* all workers eventually respond, in order to get convergence (which is Assumption 1.1 in Chap. 6 of (Bertsekas and Tsitsiklis, 1989) for totally asynchronous algorithms).

8.2.2 Preliminary: local iterations

To understand why the algorithm converges as a whole, let us first take a close look at how one local iteration of RPG enables iterates to get closer to a local solution. Indeed, a special property of the algorithm is that local variables (x_k^i) do not converge to the same value as the coordinator variable $\overline{x}_k$. In contrast, they go to the *local shifted optimal point* $x^{\star@i} := x^\star - \gamma^i \nabla f^i(x^\star)$.

At worker i and time k, $x_k^i = x_{k-d_k^i}^i$ was obtained by $p = p(i, k - d_k^i)$ repetitions of proximal gradient. Starting with the reception of $\overline{x}_{k-D_k^i}$ and initializing $\Delta_{(0)} = 0$, the p local iterations (indexed by subscripts with parentheses) are obtained by

$$\begin{aligned} u_{(q)} &= \mathrm{prox}_{\gamma g}(\overline{x}_{k-D_k^i} + \Delta_{(q-1)}), \\ x_{(q)}^i &= u_{(q)} - \gamma^i \nabla f^i(u_{(q)}) \\ \Delta_{(q)} &= \Delta_{(q-1)} + \pi^i \left(x_{(q)}^i - x_{(q-1)}^i \right) \end{aligned}$$

for $q = 1, .., p$. Then, $x_{k-D_k^i}^i = x_{(p)}^i$ and $\Delta_{k-d_k^i} = \Delta_{(p)}$.

The next lemma is fundamental to the analysis of our algorithm. It describes how the local computations go towards their own local shifted optimal point, compared to

$$\mathrm{a}_k := \max\left(\left\| \overline{x}_k - \overline{x}^\star \right\|^2, \left\| \overline{x}_k^{\setminus i_k} - \overline{x}^{\star \setminus i_k} \right\|^2 \right),$$

where i_k is the updating agent at time k and

$$\overline{x}^\star = \sum_{i=1}^M \pi^i x^{\star@i}, \qquad \overline{x}_k^{\setminus i} = \frac{1}{1-\pi^i} \sum_{j \neq i} \pi^j x_k^j, \qquad \overline{x}^{\star \setminus i} = \frac{1}{1-\pi^i} \sum_{j \neq i} \pi^j x^{\star@j}.$$

In addition, we have that $x^\star = \mathrm{prox}_{\gamma g}(\overline{x}^\star)$ by first-order optimality conditions of Problem ($\mathcal{P}_{\mathrm{B-8}}$).

Lemma 8.2. *Let f^i be convex and L^i-smooth, g be convex lower semi-continuous. Then, with $\gamma^i \in (0, 2/L^i)$, we have for any k and any number of repetitions*

$$\left\|x_k^i - x^{\star@i}\right\|^2 \leq \mathrm{a}_{k-D_k^i} - \gamma^i \left(\frac{2}{L^i} - \gamma^i\right) \left\|\nabla f^i(u_{(p)}) - \nabla f^i(x^\star)\right\|^2$$

where $u_{(p)}$ is such that $x_k^i = u_{(p)} - \gamma^i \nabla f^i(u_{(p)})$.
Furthermore, if f^i is additionally μ^i-strongly convex. Then, with $\gamma^i \in (0, 2/(L^i + \mu^i)]$, we have for any k that after p_k^i repetitions

$$\left\|x_k^i - x^{\star@i}\right\|^2 \leq \left(1 - \gamma^i\mu^i\right)^2 g^i(p_k^i)^2 \, \mathrm{a}_{k-D_k^i}$$

with $g^i(p) = 1 - \gamma^i\mu^i \sum_{q=1}^{p-1} (1 - \gamma^i\mu^i)^{q-1} (\pi^i)^q$.

Proof. First, as f^i is μ^i-strongly convex and L^i smooth, we have that for any $q = 1, .., p$ (see Eq. (2.13) in Lemma 2.34),

$$\begin{aligned}
&\left\|x_{(q)} - x^{\star@i}\right\|^2 = \left\|u_{(p)} - \gamma^i \nabla f^i(u_{(q)}) - (x^\star - \gamma^i \nabla f^i(x^\star))\right\|^2 \\
&\leq \left(1 - \frac{2\gamma^i\mu^i L^i}{\mu^i + L^i}\right) \left\|u_{(q)} - x^\star\right\|^2 - \gamma^i \left(\frac{2}{\mu^i + L^i} - \gamma^i\right) \left\|\nabla f^i(u_{(q)}) - \nabla f^i(x^\star)\right\|^2 \quad (8.6) \\
&\leq \left[\left(1 - \frac{2\gamma^i\mu^i L^i}{\mu^i + L^i}\right) - \mu^2\gamma^i \left(\frac{2}{\mu^i + L^i} - \gamma^i\right)\right] \left\|u_{(q)} - x^\star\right\|^2 \\
&= (1 - \gamma^i\mu^i)^2 \left\|u_{(q)} - x^\star\right\|^2 . \quad (8.7)
\end{aligned}$$

Then, for $q = 1$, we have by non-expansivity of the proximity operator that

$$\left\|u_{(1)} - x^\star\right\|^2 \leq \left\|\overline{x}_{k-D_k^i} - \overline{x}^\star\right\|^2$$

which completes the proof for $p = 1$. Going further, for $q \geq 2$, non-expansivity and Jensen's inequality yield

$$\begin{aligned}
\left\|u_{(q)} - x^\star\right\|^2 &\leq \left\|\overline{x}_{k-D_k^i} + \Delta_{(q-1)} - \overline{x}^\star\right\|^2 \\
&= \left\|\pi^i \left(x_{(q-1)} - x^{\star@i}\right) + \sum_{j \neq i} \pi^j \left(x^j_{k-D_k^i} - x^{\star@j}\right)\right\|^2 \\
&= \left\|\pi^i \left(x_{(q-1)} - x^{\star@i}\right) + (1 - \pi^i) \left(\overline{x}^{\setminus i}_{k-D_k^i} - \overline{x}^{\star\setminus i}\right)\right\|^2 \\
&\leq \pi^i \left\|x_{(q-1)} - x^{\star@i}\right\|^2 + (1 - \pi^i) \left\|\overline{x}^{\setminus i}_{k-D_k^i} - \overline{x}^{\star\setminus i}\right\|^2 .
\end{aligned}$$

Then by induction, using the triangle inequality instead of convexity, one gets that

for $p \geq 2$ (and using $v^i = (1-\gamma^i\mu^i)\pi^i$)

$$\begin{aligned}
\left\|u_{(p)} - x^\star\right\| &\leq \pi^i \left\|x_{(p-1)} - x^{\star@i}\right\| + (1-\pi^i)\left\|\overline{x}^{\backslash i}_{k-D^i_k} - \overline{x}^{\star\backslash i}\right\| \\
&\leq v^i \left\|u_{(p-1)} - x^\star\right\| + (1-\pi^i)\sqrt{a_{k-D^i_k}} \\
&\leq v^i_{(p-1)} \left\|u_{(1)} - x^\star\right\| + \left[\sum_{q=1}^{p-1} v^i_{(q-1)}(1-\pi^i)\right]\sqrt{a_{k-D^i_k}} \\
&\leq v^i_{(p-1)} \left\|\overline{x}_{k-D^i_k} - x^\star\right\| + \left[\sum_{q=1}^{p-1} v^i_{(q-1)}(1-\pi^i)\right]\sqrt{a_{k-D^i_k}} \\
&\leq v^i_{(p-1)}\sqrt{a_{k-D^i_k}} + \left[\sum_{q=1}^{p-1} v^i_{(q-1)}(1-\pi^i)\right]\sqrt{a_{k-D^i_k}} \\
&= \left[v^i_{(p-1)} + \sum_{q=0}^{p-2} v^i_{(q)} - \frac{1}{1-\gamma^i\mu^i}\sum_{q=1}^{p-1} v^i_{(q)}\right]\sqrt{a_{k-D^i_k}} \\
&= \underbrace{\left[1 - \frac{\gamma^i\mu^i}{1-\gamma^i\mu^i}\sum_{q=1}^{p-1} v^i_{(q)}\right]}_{:=g^i(p)}\sqrt{a_{k-D^i_k}}
\end{aligned}$$

noting that $i = i_{k-D^i_k}$ was updating at time $k-D^i_k$ by definition. Using the last inequality on top of (8.6) or (8.7) leads to the claim, noting that $g^i(p) = 1$ for all p when $\mu^i = 0$. □

8.2.3 Convergence results

In this section, we analyze the convergence of our algorithm in the strongly convex case.[59] Our results allow us to choose the same stepsize as for vanilla gradient descent (without any dependence on the delays). The derived rates involve the *number of epochs* rather than the number of iterations. In Section 8.2.4, we will examine how these rates translate in terms of number of iteration when the delays are bounded in order to compare with the literature.

[59] Lemma 8.2 also enables to show the convergence and a sublinear rate in the (non-strongly) convex case as presented in (Mishchenko et al., 2020) and discussed quickly in Section 8.3.1.

Theorem 8.3. *Let Assumption 8.1 hold. Using $\gamma^i \in (0, \frac{2}{\mu^i+L^i}]$, DAve-RPG (Algorithm 8.1) converges linearly to the solution of ($\mathcal{P}_{B-8}$) on the epoch sequence (k_ℓ). More precisely, for all $k \in [k_\ell, k_{\ell+1})$*

$$\left\|x_k - x^\star\right\|^2 \leq \left(1 - \min_i \gamma^i\mu^i\right)^{2\ell} \max_i \left\|x^i_0 - x^{\star@i}\right\|^2,$$

with the shifted local solutions $x^{\star@i} = x^\star - \gamma^i\nabla f^i(x^\star)$.

Proof. First, for any i and any $k \in [k_\ell, k_{\ell+1})$, we have from Lemma 8.2

$$\left\|x^i_k - x^{\star@i}\right\|^2 = (1-\gamma^i\mu^i)^2 g^i(p^i_k)^2 \ a_{k-D^i_k} \leq (1-\beta)^2 \ a_{k-D^i_k},$$

with $\beta := \min_i \gamma^i \mu^i$. Thus, for any $k \in [k_\ell; k_{\ell+1})$,

$$\left\|\overline{x}_k - \overline{x}^\star\right\|^2 \leq \sum_{i=1}^M \pi^i \|x_k^i - x^{\star@i}\|^2 = \sum_{i=1}^M \pi^i \|x_{k-d_k^i}^i - x^{\star@i}\|^2$$
$$\leq (1-\beta)^2 \sum_{i=1}^M \pi^i \mathrm{a}_{k-D_k^i} \leq (1-\beta)^2 \max_i \mathrm{a}_{k-D_k^i}$$

Similarly, for any j

$$\left\|\overline{x}_k^{\setminus j} - \overline{x}^{\star\setminus j}\right\|^2 \leq (1-\pi^j)^{-1} \sum_{i\neq j} \pi^i \left\|x_{k-d_k^i}^i - x^{\star@i}\right\|^2 \leq (1-\beta)^2 \max_i \mathrm{a}_{k-D_k^i}.$$

Finally, we get

$$\mathrm{a}_k \leq (1-\beta)^2 \max_i \mathrm{a}_{k-D_k^i}$$

which is the workhorse for the rest of the proof.

Let $\ell > 0$ and $k \in [k_\ell, k_{\ell+1})$, then the definition of the epoch sequence (k_ℓ) gives $k - D_k^i \geq k_{\ell-1}$ and then

$$\mathrm{a}_k \leq (1-\beta)^2 \max_i \mathrm{a}_{k-D_k^i} \leq (1-\beta)^2 \max_{k' \in [k_{\ell-1}, k)} \mathrm{a}_{k'}$$

and applying this inequality sequentially to $k_\ell, k_\ell + 1, \ldots, k_{\ell+1} - 1$, we get

$$\begin{aligned}
\mathrm{a}_{k_\ell} &\leq (1-\beta)^2 \max_{k' \in [k_{\ell-1}, k_\ell)} \mathrm{a}_{k'}, \qquad (8.8)\\
\mathrm{a}_{k_\ell+1} &\leq (1-\beta)^2 \max\left(\max_{k' \in [k_{\ell-1}, k_\ell)} \mathrm{a}_{k'}, \mathrm{a}_{k_\ell}\right)\\
&\leq (1-\beta)^2 \max_{k' \in [k_{\ell-1}, k_\ell)} \mathrm{a}_{k'} \quad \text{(using Eq. (8.8))}\\
&\ldots\\
\max_{k \in [k_\ell, k_{\ell+1})} \mathrm{a}_k &\leq (1-\beta)^2 \max_{k' \in [k_{\ell-1}, k_\ell)} \mathrm{a}_{k'}\\
&\leq (1-\beta)^{2\ell} \max_{k' < k_0} \mathrm{a}_{k'} \leq (1-\beta)^{2\ell} \max_i \left\|x_0^i - x^{\star@i}\right\|^2.
\end{aligned}$$

Finally, since the proximity operator of a convex function is non-expansive, we have for all $k \in [k_\ell; k_{\ell+1})$,

$$\|x_k - x^\star\|^2 = \|\mathrm{prox}_{\gamma g}(\overline{x}_k) - \mathrm{prox}_{\gamma g}(\overline{x}^\star)\|^2 \leq \|\overline{x}_k - \overline{x}^\star\|^2$$
$$\leq \max_{k \in [k_\ell, k_{\ell+1})} \mathrm{a}_k \leq (1-\beta)^{2\ell} \max_i \left\|x_0^i - x^{\star@i}\right\|^2$$

which concludes the proof. □

Notice that the rate provided by this theorem is valid for any choice of number of local iterations at any worker/time. The local contraction at agent i can indeed be improved by doing p local repetitions by a factor

$$r^i(p) = 1 - \gamma^i \mu^i \sum_{q=1}^{p-1} (1-\gamma^i\mu^i)^{q-1} (\pi^i)^q = 1 - \gamma^i \mu^i \pi^i \frac{1 - (1-\gamma^i\mu^i)^{p-1}(\pi^i)^{p-1}}{1 - (1-\gamma^i\mu^i)\pi^i}$$

where $r^i(1) = 1$ and r^i is decreasing with p and lower-bounded by

$$r^i(\infty) = 1 - \frac{\gamma^i \mu^i \pi^i}{1 - (1 - \gamma^i \mu^i)\pi^i}.$$

If all workers, or at least the ones with the slowest rates, perform several local iterations, the rate can thus be improved from $(1 - \min_i \gamma^i \mu^i)^2$ to

$$\max_{i,k} \left(1 - \gamma^i \mu^i\right)^2 r^i(p_k^i)^2.$$

However, local iterations practically slow down the actual time between two epochs thus the number of local repetitions have to be carefully tuned in practice. The flexibility allowed by our algorithm enables a wide range of selection strategies such as online tuning, stopping the local iterations after some fixed time, etc.

8.2.4 Comparison of the results with the literature

The main feature of the epoch sequence introduced in Section 8.2.1 is that it automatically adapts to variations of behaviors of machines across time (such as one worker being slow at first that gets faster with time). The sequence then allows for a intrinsic convergence analysis without any knowledge of the delays, as shown in the previous sections. This simple but powerful remark is one of the main technical contributions of this paper.

For comparisons with the literature, the following result provides explicit connections between number of iterations and number of epochs with two standard bounds on delays uniformly in time.

[60] *For $M = 1$ machine, we have $k_\ell = \ell$ as mentioned in Section 8.2.1 and we recover exactly the convergence rates of the vanilla proximal gradient.*

Proposition 8.4 (epoch scaling with delays). *For $M > 1$ machines,[60] uniformly over time:*

- *if the delays are uniformly bounded by d over the workers,* i.e. $d_k^i \leq d$ *for all i, then $d \geq M$ and the epoch sequence has complexity $k_\ell = O(\ell M)$;*
- *if the average delay is bounded by $\overline{d}$,* i.e. $1/M \sum_{i=1}^M d_k^i \leq \overline{d}$, *then $\overline{d} \geq (M-1)/2$ and the epoch sequence has complexity $k_\ell = O(\ell M)$.*

	uniform bound	average bound
Condition	$d_k^i \leq d$ for all i	$\frac{1}{M}\sum_{i=1}^M d_k^i \leq \overline{d}$
Unimprov. bound	$d = M + \tau; \quad \tau \geq 0$	$\overline{d} = \frac{M-1}{2} + \tau; \quad \tau \geq 0$
1 Epoch	$k_{\ell+1} - k_\ell \leq 2d + 1$	$k_{\ell+1} - k_\ell \leq 2M(2\overline{d} - M + 3) - 3$
Epoch sequence	$k_\ell \leq (2M + 2\tau + 1)\ell$	$k_\ell \leq 4M(\tau + 1)\ell$

Bounding the average delay among the workers is an attractive assumption which is however much less common in the literature. The defined epoch sequence and associated analysis subsumes this kind of assumption.

In the case of uniformly bounded delays, the derived link between epoch and time sequence enables us to compare our rates in the strongly convex case (Theorem 8.3) with the ones obtained for PIAG (Aytekin et al., 2016; Vanli et al., 2016, 2018). To simply the comparison, let us consider the case where all the workers share the same strong convexity and smoothness constants μ and L. The first thing to notice is that

the admissible stepsize for PIAG depends on the delays' uniform upper bound d which is practically concerning, while the usual proximal gradient stepsizes are used for the proposed DAve-RPG. Using the optimal stepsizes in each case, the convergence rates in terms of time k are:

	DAve-RPG	PIAG
Reference	Th. 8.3	Th. 3.4 of (Vanli et al., 2016)
Stepsize	$\gamma = \frac{2}{\mu+L}$	$\gamma = \frac{16}{\mu}\left[(1+\frac{\mu}{48L})^{\frac{1}{d+1}} - 1\right]$
Rate	$\left(1 - \frac{2}{1+\frac{L}{\mu}}\right)^{\frac{k}{d+0.5}}$	$\left(1 - \frac{1}{49\frac{L}{\mu}}\right)^{\frac{k}{d+1}}$

We notice in both cases the exponent inversely proportional to the maximal delay d but the term inside the parenthesis is a hundred times closer to 1 for PIAG. Even if our algorithm is made for handling the flexible delays, this comparison illustrates the interest of our approach over PIAG for distributed asynchronous optimization in the case of bounded delays.

8.3 Extensions and Further developments

8.3.1 About the proof techniques & assumptions

The proof above illustrates our technique for encompassing delays without treating them as noise. What we lose in counterpart is that we only consider one update per agent and epoch while there could be many more. However, since in one epoch the slowest agent updates exactly twice, we claim that this loss is reasonable with respect to the whole objective.

The proof is heavily based on maintaining the average of the users contribution and then directly working on the local improvements of the users towards their shifted minimizers. This work on the iterates is inspired by the monotone operators' theory, and thus necessitates convexity. However, strong convexity is not required for convergence. We have shown the following result in (Mishchenko et al., 2020).

Theorem 8.5. *Let the (f^i) be convex L^i-smooth, g be convex lower semi-continuous, and $\gamma^i \in (0, 2/L^i)$. Then, if $x^\star$ is the unique minimizer of $(\mathcal{P}_{\mathrm{B-8}})$, the sequence (x_k) converges to $x^\star$. Moreover, if Problem $(\mathcal{P}_{\mathrm{B-8}})$ has multiples minimizers, then (x_k) still converges to a minimizer of $(\mathcal{P}_{\mathrm{B-8}})$, under two additional assumptions: (i) the difference between two consecutive epochs $k_\ell - k_{\ell-1}$ is uniformly bounded; and (ii) the number of inner loops is uniformly bounded.*
Furthermore, for any $k \in [k_\ell, k_{\ell+1})$, we have

$$\min_{k' \leq k} \|\partial F(x_{k'})\| \leq \frac{2\sqrt{2}}{\sqrt{\ell}} \frac{\max_i \left\|x_0^i - x^{\star@i}\right\|}{\min_j \left(\gamma^j \sqrt{2 - \gamma^j L^j}\right)},$$

where $\|\partial F(x_{k'})\| \coloneqq \min_{h \in 1/M \sum_i \nabla f^i(x_{k'}) + \partial g(x_{k'})} \|h\|$.

8.3.2 The special case of similar functions

Let us now focus on the case when the smooth functions F^i are similar in the sense that they have similar conditioning. In this case, we can rescale them so that they have

similar smoothness and strong convexity parameters.

The typical application we can have in mind is when the agents have empirical losses of the form $\sum_{j\in\mathcal{S}_m^i} \ell(b_j, P_x(a_j))$ on a local dataset $\mathcal{S}_m^i$.

Thus, we will now consider the case where the workers functions are of the form

$$F^i(x) = \alpha^i \left(f^i(x) + g(x)\right)$$

with $\alpha^i \in (0,1)$ and $\sum_{i=1}^M \alpha^i = 1$. This leads to a slightly different global objective

$$\min_{x\in\mathbb{R}^n} F(x) := \sum_{i=1}^M F^i(x) = \sum_{i=1}^M \alpha^i f^i(x) + g(x) \qquad (\mathcal{P}'_{\text{B}-8})$$

and a different kind of assumption with common smoothness and strong convexity parameters.

Assumption 8.6. For each $i = 1, .., M$, the function $f^i : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex and L-smooth; the function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is convex, proper, and lower semi-continuous.

Then, Algorithm 8.1 can be quite directly changed by replacing the π^i by α^i and taking the same stepsize everywhere.

Algorithm 8.2 DAve-RPG for $(\mathcal{P}'_{\text{B}-8})$

Coordinator:

Initialize $\overline{x} = \overline{x}_0$, $k = 0$
while test C not verified **do**
  Receive adjustment Δ_k from worker i_k
  $\overline{x}_k = \overline{x}_{k-1} + \Delta_k$
  Send $\overline{x}_k$ to the worker in return
  $k \leftarrow k + 1$
end while
Interrupt all workers
Output $x_k = \text{prox}_{\gamma g}(\overline{x}_k)$

Worker i:

Initialize $x = x^i = \overline{x}$
while not interrupted by coordinator **do**
  Receive $\overline{x}$ from the coordinator
  Select a number of repetitions p
  $\Delta \leftarrow 0$
  for $q = 1, .., p$ **do**
    $u \leftarrow \text{prox}_{\gamma g}(\overline{x} + \Delta)$
    $y \leftarrow u - \gamma \nabla f^i(u)$
    $\Delta \leftarrow \Delta + \alpha^i (y - x)$
    $x \leftarrow y$
  end for
  Send adjustment Δ to the coordinator
end while

Then, Lemma 8.2 and Theorem 8.3 can be slightly modified to get the following result (see (Mishchenko et al., 2018) for details).

Theorem 8.7. *Let Assumption 8.6 hold. Using $\gamma \in (0, \frac{2}{\mu+L}]$, DAve-RPG (Algorithm 8.2) converges linearly to the solution of $(\mathcal{P}'_{\text{B}-8})$ on the epoch sequence (k_ℓ). More precisely, for all $k \in [k_\ell, k_{\ell+1})$*

$$\left\|x_k - x^\star\right\|^2 \leq (1-\gamma\mu)^{2\ell} \max_i \left\|x_0^i - x^{\star@i}\right\|^2,$$

with the shifted local solutions $x^{\star@i} = x^\star - \gamma \nabla f^i(x^\star)$.

8.3.3 On the communications

Communications are of utmost importance in computing networks. In the previous sections, we insisted on the fact that asynchronous exchanges can be managed. Now, the communications can also be reduced.

Scarse communications

We actually have the additional feature that multiple "local steps" can be performed before exchanging with the coordinator. This means that our method has the possibility to adapt not too often and this will even improve the actual rate in terms of epochs. However, the gain is not clear in terms of wallclock time is highly system-dependent (since the epochs durations are longer).

Sparse communications

We saw in Part A that proximity operators could bring structure (*e.g.* sparsity) to the iterates. This structure could then be used to encode the vectors communicated between the coordinator and some worker. Unfortunately, in either Algorithm 8.1 or Algorithm 8.2, the "gradients" are communicated, not the proximity operator output.

This can be remediated when we restrict ourselves to 1 local repetition. Indeed, in that case, the "average point" $\overline{x}$ is not needed anymore on the worker side, only $\operatorname{prox}_{\gamma g}(\overline{x})$ is required (α^i is not need either on the worker side). We display this modification in Algorithm 8.3.

This way, the communications from the coordinator to the worker will be become structured as described in Chapter 3 as per the following result.

Theorem 8.8. *Let Assumption 8.6 hold. Suppose that* $x^\star$ *belongs to some manifold* $\mathcal{M}$ *and that* (x_k) *is a sequence produced by DAve-PG (Algorithm 8.3) with* $\gamma \in (0, \frac{2}{\mu+L}]$. *If:*

1) there is $\varepsilon > 0$ *such that*

$$\text{for all } y \in \mathcal{B}(\overline{x}^\star, \varepsilon), \quad \operatorname{prox}_{\gamma g}(y) \in \mathcal{M}, \tag{PQC}$$

or 2) g *is partly-smooth relative to* $\mathcal{M}$ *at* $x^\star$ *and*

$$0 \in \operatorname{ri} \partial F(x^\star) \tag{SQC}$$

then, after some finite time number of epochs, $x_k \in \mathcal{M}$.

Proof. By observing the end of the proof of Theorem 8.3, we observe that $(\overline{x}_k)$ converges to $\overline{x}^\star$ as ℓ goes to infinity and that $x_k = \operatorname{prox}_{\gamma g}(\overline{x}_k)$. The result with the first set of assumptions is then a direct application of Theorem 3.2 while the second set of assumptions comes from Corollary 3.15. □

Algorithm 8.3 DAve-PG for ($\mathcal{P}'_{\text{B-8}}$)

Coordinator:

Initialize $\overline{x} = \overline{x}_0$, $k = 0$
while test C not verified **do**
  Receive adjustment Δ_k from worker i_k
  $\overline{x}_k = \overline{x}_{k-1} + \alpha^{i_k}\Delta_k$
  $x_k = \text{prox}_{\gamma g}(\overline{x}_k)$
  Send x_k to the worker in return
  $k \leftarrow k+1$
end while
Interrupt all workers
Output x_k

Worker i:

Initialize $x = x^i$
while not interrupted by coordinator **do**
  Receive x from the coordinator
  $y \leftarrow x - \gamma \nabla f^i(x)$
  $\Delta \leftarrow y - x^i$
  $x^i \leftarrow y$
  Send adjustment Δ to the coordinator
end while

Thus, by encoding appropriately the exchanges from the coordinator to the workers (typically by encoding for sparse vectors), the communication cost can be reduced without any performance loss.

A natural follow-up is thus whether the communication from the workers to the coordinator can also be reduced by some form sparsification, keeping in mind the adaptive sparsification concepts from Chapter 5.

8.4 Exchanges reduction for sparse problems

In this section, we will focus on *sparsity inducing* nonsmooth functions for g.[61]

[61] As in Chapter 5, our results do not rely on separability and can thus encompass various functions.

8.4.1 Random sparsification of the workers updates

A direct approach to reduce the worker to coordinator exchanges in DAve-PG is to update only a random subset of the coordinates. Mathematically, at iteration k, the random subset of entries that worker i_k updates is denoted by $\boldsymbol{S}_{k-D_k^i}$ (in bold, emphasizing that it is the only random variable in the algorithm, the time index $k - D_k^i$ meaning that it is drawn just after $x_{k-D_k^i}$ is computed). The update writes

$$x_k^{i[j]} = \begin{cases} \left(x_{k-D_k^i} - \gamma \nabla f^i(x_{k-D_k^i})\right)^{[j]} & \text{if } \left| \begin{array}{l} i = i_k \\ j \in \boldsymbol{S}_{k-D_k^i} \end{array} \right. \\ x_{k-1}^{i[j]} & \text{otherwise} \end{cases}$$

$$x_k = \text{prox}_{\gamma g}(\overline{x}_k) \qquad \text{with} \quad \overline{x}_k = \sum_{i=1}^{M} \alpha^i x_k^i,$$

where $x_k^{i[j]}$ denotes the j coordinate of x local at worker i at time k.

With this sparsification, the local updates correspond to a random block coordinate descent step for the workers. However, this algorithm does not boil down to an asynchronous stochastic block-coordinate descent algorithm such as (Liu et al., 2015;

Peng et al., 2016; Richtárik and Takáč, 2016a; Sun et al., 2017), since our method maintains a variable, $\overline{x}_k$, aggregating asynchronously all the workers' contributions.

Assumption 8.9 (On the random sparsification)**.** The sparsity selectors (S_k) are independent and identically distributed random variables. We select a coordinate in S_k as follows:

$$\mathbb{P}[j \in S_k] = p_j > 0 \quad \text{for all } j \in \{1, .., n\},$$

with $\mathrm{p} = (p_1, .., p_n) \in (0, 1]^n$. We denote $\mathrm{p}_{\max} = \max_i p_i$ and $\mathrm{p}_{\min} = \min_i p_i$.

The selectors (S_k) being the only random variables of the algorithm, it is natural to define the filtration $\mathcal{F}_k = \sigma(\{S_{k'}\}_{k'<k})$ so that all variables at time k (x_k^i, $\overline{x}_k$, x_k, d_k^i, D_k^i) are $\mathcal{F}_k$-measurable but S_k is not.

Using this sparsification DAve-PG becomes Algorithm 8.4, that we call DAve-SPG. Similarly to DAve-PG, none of the ingredient of method, including the stepsize choice, depend on the computing system. It also shares the feature that although each coordinator update involves only one worker (and thus part of the data), all the data is always implicitly involved in the coordinator variable; which allows the algorithm to cope with the heterogeneity of the computing system (data distribution, workers delays).[62]

[62] We recall our notation: for a vector of $x \in \mathbb{R}^n$ and a subset S of $\{1, .., n\}$, $[x]^S$ denotes the sparse size-n vector where S is the set of non-null entries, for which they match those of x, *i.e.* $([x]^S)^{[j]} = x^{[j]}$ if $j \in S$ and 0 otherwise.

Algorithm 8.4 DAve-SPG on $((\alpha^i), (f^i), g\ ;\ \mathrm{p})$ with stopping criterion C

Coordinator:

Initialize $\overline{x} = \overline{x}_0$, $k = 0$
while test C not verified **do**
  Receive adjustment $[\Delta_k]^{S_{k-D_k^{i_k}}}$ from worker i_k
  $\overline{x}_k \leftarrow \overline{x}_{k-1} + \alpha^{i_k} [\Delta_k]^{S_{k-D_k^{i_k}}}$
  $x_k = \mathrm{prox}_{\gamma g}(\overline{x}_k)$
  Draw sparsity S_k with prob. p
  Send x_k, S_k to the worker in return
  $k \leftarrow k + 1$
end while
Interrupt all workers
Output x_k

Worker i:

Initialize $x = x^i$
while not interrupted by coordinator **do**
  Receive x and S from the coordinator
  $[y]^S \leftarrow [x - \gamma \nabla f^i(x)]^S$
  $\Delta \leftarrow y - x^i$
  Send adjustment Δ to the coordinator
  $[x^i]^S \leftarrow [y]^S$
end while

The communications per iteration are (i) a blocking send/receive from a worker to the coordinator (in blue) of size $|S|$, and (ii) a blocking send/receive from the coordinator to the last updating worker (in orange) of the current iterate. The worker-to-coordinator communications are thus made sparse by the algorithm and the coordinator-to-worker communications costs depend on the structure of x_k, which is the output of a proximal operator on g as previously discussed.

8.4.2 Convergence Analysis & Limits of Sparsification

By incorporating the random sparsification procedure of DAve-SPG in the proofs above, we obtain the following result.

Theorem 8.10 (Reaches and Limits of Sparsification)**.** *Let Assumption 8.6 hold. Take* $\gamma \in (0, \frac{2}{\mu+L}]$. *Suppose that Assumption 8.9 holds for the probability vector* p *with* $\frac{\mathrm{p}_{\min}}{\mathrm{p}_{\max}} \geq (1-\gamma\mu)^2$.
Then, DAve-SPG on $((\alpha^i), (f^i), g\,;\,\mathrm{p})$ *verifies for all* $k \in [k_\ell, k_{\ell+1})$

$$\mathbb{E}\left\|x_k - x^\star\right\|^2 \leq \left(\mathrm{p}_{\max}\left(\frac{1-\kappa_{(\mathcal{P}'_{\mathrm{B-8}})}}{1+\kappa_{(\mathcal{P}'_{\mathrm{B-8}})}}\right)^2 + 1 - \mathrm{p}_{\min}\right)^\ell C \quad \text{for maximal } \gamma = \frac{2}{\mu+L} \tag{8.9}$$

$$\leq \left(\mathrm{p}_{\max}(1-\gamma\mu)^2 + 1 - \mathrm{p}_{\min}\right)^\ell C \quad \text{for any } \gamma \in \left(0, \frac{2}{\mu+L}\right]. \tag{8.10}$$

with $C = \max_i \left\|x_0^i - x^{\star@i}\right\|^2$ *and* $x^{\star@i} = x^\star - \gamma\nabla f^i(x^\star)$.

This result establishes bounds that lead to convergence whenever the selection probabilities are well chosen. First, if all probabilities are equal to 1, the algorithm boils down to DAve-PG and Theorem 8.10 coincides with Theorem 8.7. In more general cases, this result has to be interpreted more carefully as developed next.

Proof. From the solution $x^\star$ of $(\mathcal{P}'_{\mathrm{B-8}})$ (unique from strong convexity), we define (as before) for each worker i the local shift $x^{\star@i} = x^\star - \gamma\nabla f^i(x^\star)$, as well as their average $\overline{x}^\star = \sum_{i=1}^M \alpha^i x^{\star@i}$. First-order optimality conditions $0 \in \sum_i \alpha^i \nabla f^i(x^\star) + \partial g(x^\star)$ imply that

$$\overline{x}^\star = \sum_{i=1}^M \alpha^i x^{\star@i} = x^\star - \gamma \sum_{i=1}^M \alpha^i \nabla f^i(x^\star) \;\in\; x^\star + \gamma\partial g(x^\star)$$

which directly leads to $\mathrm{prox}_{\gamma g}(\overline{x}^\star) = x^\star$.

Note now that, for a time k and a worker i, we have that $x_k^i = x^i_{k-d_k^i}$ depends on $x_{k-D_k^i}$ (which is $\mathcal{F}_{k-D_k^i}$-measurable) and on $S_{k-D_k^i}$ (which is i.i.d.). First, we control the term $\|x_k^i - x^{\star@i}\|^2$.

Let us define $\|x\|_{\mathrm{p}}^2 = \sum_{j=1}^n p_j (x^{[j]})^2$ where $(p_1, .., p_n)$ is the vector of probabilities of Assumption 8.9. The conditional expectation can be developed as follows:

$$\begin{aligned}\mathbb{E}[\|x_k^i - x^{\star@i}\|^2 | \mathcal{F}_{k-D_k^i}] &= \mathbb{E}[\|x^i_{k-d_k^i} - x^{\star@i}\|^2 | \mathcal{F}_{k-D_k^i}] = \sum_{j=1}^n \mathbb{E}[(x^{i[j]}_{k-D^{i}k} - x^{\star@i[j]})^2 | \mathcal{F}_{k-D_k^i}] \\ &= \|x_{k-D_k^i} - \gamma\nabla f^i(x_{k-D_k^i}) - (x^\star - \gamma\nabla f^i(x^\star))\|_{\mathrm{p}}^2 + \|x^i_{k-D_k^i} - x^{\star@i}\|_{1-\mathrm{p}}^2.\end{aligned}$$

Let us now bound both terms of this sum.

$$\begin{aligned}&\|x_{k-D_k^i} - \gamma\nabla f^i(x_{k-D_k^i}) - (x^\star - \gamma\nabla f^i(x^\star))\|_{\mathrm{p}}^2 + \|x^i_{k-D_k^i} - x^{\star@i}\|_{1-\mathrm{p}}^2 \\ &\quad \leq \mathrm{p}_{\max}\|x_{k-D_k^i} - \gamma\nabla f^i(x_{k-D_k^i}) - (x^\star - \gamma\nabla f^i(x^\star))\|^2 + (1-\mathrm{p}_{\min})\|x^i_{k-D_k^i} - x^{\star@i}\|^2.\end{aligned}$$

We now use the μ-strong convexity and L-smoothness of f^i (Lemma 2.34) to write

$$\begin{aligned}
&\|x_{k-D_k^i} - \gamma\nabla f^i(x_{k-D_k^i}) - (x^\star - \gamma\nabla f^i(x^\star))\|^2 \\
&\quad\le \left(1 - \frac{2\gamma\mu L}{\mu+L}\right)\left\|x_{k-D_k^i} - x^\star\right\|^2 - \gamma\left(\frac{2}{\mu+L} - \gamma\right)\left\|\nabla f^i(x_{k-D_k^i}) - \nabla f^i(x^\star)\right\|^2 \\
&\quad\le \left[\left(1 - \frac{2\gamma\mu L}{\mu+L}\right) - \mu^2\gamma\left(\frac{2}{\mu+L} - \gamma\right)\right]\left\|x_{k-D_k^i} - x^\star\right\|^2 = (1-\gamma\mu)^2\left\|x_{k-D_k^i} - x^\star\right\|^2
\end{aligned}$$

for any $\gamma \in (0, 2/(\mu+L)]$. Thus,

$$\begin{aligned}
\mathbb{E}[\|x_k^i - x^{\star@i}\|^2|\mathcal{F}_{k-D_k^i}] &\le \mathrm{p}_{\max}(1-\gamma\mu)^2\left\|x_{k-D_k^i} - x^\star\right\|^2 + (1-\mathrm{p}_{\min})\|x_{k-D_k^i}^i - x^{\star@i}\|^2 \\
&\le \mathrm{p}_{\max}(1-\gamma\mu)^2\left\|\overline{x}_{k-D_k^i} - \overline{x}^\star\right\|^2 + (1-\mathrm{p}_{\min})\|x_{k-D_k^i}^i - x^{\star@i}\|^2,
\end{aligned}$$

where we used that $\|x_{k-D_k^i} - x^\star\|^2 = \|\mathrm{prox}_{\gamma g}(\overline{x}_{k-D_k^i}) - \mathrm{prox}_{\gamma g}(\overline{x}^\star)\|^2 \le \|\overline{x}_{k-D_k^i} - \overline{x}^\star\|^2$ by definition and non-expansiveness of the proximity operator of g.

Taking full expectation on both sides and using $\overline{x}_{k-D_k^i} - \overline{x}^\star = \sum_{i=1}^M \alpha^i(x_{k-D_k^i}^i - \overline{x}^{\star@i})$, we get

$$\begin{aligned}
\mathbb{E}\|x_k^i - x^{\star@i}\|^2 &\le \mathrm{p}_{\max}(1-\gamma\mu)^2\sum_{j=1}^M \alpha^j\mathbb{E}\left\|x_{k-D_k^i}^j - x^{\star@j}\right\|^2 + (1-\mathrm{p}_{\min})\mathbb{E}\|x_{k-D_k^i}^i - x^{\star@i}\|^2 \\
&\le \mathrm{p}_{\max}(1-\gamma\mu)^2\max_{j=1,..,M}\mathbb{E}\left\|x_{k-D_k^i}^j - x^{\star@j}\right\|^2 + (1-\mathrm{p}_{\min})\max_{j=1,..,M}\mathbb{E}\|x_{k-D_k^i}^j - x^{\star@j}\|^2 \\
&\le \left(\mathrm{p}_{\max}(1-\gamma\mu)^2 + 1 - \mathrm{p}_{\min}\right)\max_{j=1,..,M}\mathbb{E}\left\|x_{k-D_k^i}^j - x^{\star@j}\right\|^2.
\end{aligned}$$

Let $c_k = \max_{i=1,..,M}\mathbb{E}\left\|x_k^j - x^{\star@j}\right\|^2$ and $\beta = \left(\mathrm{p}_{\max}(1-\gamma\mu)^2 + 1 - \mathrm{p}_{\min}\right)$ (note that the assumptions imply that $\beta \le 1$), then the above result implies that $c_k \le \beta\max_{j=1,..,M} c_{k-D_k^j}$ and using the definition of the sequence (k_ℓ), we get

$$\begin{aligned}
c_{k_\ell} &\le \beta\max_j c_{k_\ell - D_{k_\ell}^j} \le \beta\max_{k'\in[k_{\ell-1},k_\ell)} c_{k'} \\
c_{k_\ell+1} &\le \beta\max(c_{k_\ell}, \max_{k'\in[k_{\ell-1},k_\ell)} c_{k'}) \le \beta\max_{k'\in[k_{\ell-1},k_\ell)} c_{k'}.
\end{aligned}$$

Thus for all $k \ge k_\ell$, $c_k \le \beta\max_{k'\in[k_{\ell-1},k_\ell)} c_{k'}$. This implies that the sequence $\widetilde{c}_\ell$ defined by $\widetilde{c}_\ell = \max_{k'\in[k_{\ell-1},k_\ell)} c_{k'}$ decays exponentially: $\widetilde{c}_\ell \le \beta\widetilde{c}_{\ell-1} \le \beta^\ell\widetilde{c}_0 \le \beta^\ell\max_{i=1,..,M}\|x_0^i - x^{\star@i}\|^2$. Finally, we use again the non-expansivity of the proximity operator of g to get that for all $k \in [k_\ell, k_{\ell+1})$,

$$\mathbb{E}\|x_k - x^\star\|^2 \le \mathbb{E}\|\overline{x}_k - \overline{x}^\star\|^2 \le \sum_{i=1}^M \alpha^i\mathbb{E}\|x_k^i - x^{\star@i}\|^2 \le c_k \le \beta^\ell\max_{i=1,..,M}\|x_0^i - x^{\star@i}\|^2,$$

which concludes the proof. □

In the totally distributed setting, all machines are responsive, which means with our notation: $\ell \to \infty$ when $k \to \infty$. Then, Theorem 8.10 gives linear convergence of the mean squared error in terms of epochs if

$$\frac{\mathrm{p}_{\min}}{\mathrm{p}_{\max}} > (1-\gamma\mu)^2 \overset{\gamma=\frac{2}{\mu+L}}{\ge} \left(\frac{1-\kappa(\mathcal{P}'_{\mathrm{B-8}})}{1+\kappa(\mathcal{P}'_{\mathrm{B-8}})}\right)^2 \tag{8.11}$$

and thus the behavior of the algorithm depends if the selection is performed uniformly (and thus structure-blind) or non-uniformly.

If the selection is *uniform*, *i.e.* $p_i = p \in (0,1]$ for all i, we directly get convergence from (8.10) as the mean squared error vanishes linearly in terms of epochs with a rate $(1-p\gamma\mu(2-\gamma\mu))$, degraded compared to the $(1-\gamma\mu)^2$ rate of DAve-PG. Unfortunately, such uniform selection also results in poor performance in many cases (as illustrated in (Grishchenko et al., 2021)). Adaptivity is key for sparsifying efficiently.

We adopt here the general idea of Chapter 5 specialized for the coordinate selection: when some coordinates get null, there is some hope that they will remain null for subsequent iterations, and it is thus natural to update preferentially the non-null coordinates. Mathematically, this means that we select coordinates in the active support as follows:

$$\mathbb{P}[j \in S_k] = \begin{cases} p & \text{if } x_k^{[j]} = 0 \\ 1 & \text{if } x_k^{[j]} \neq 0 \end{cases} \qquad \text{for all } j \in \{1,..,n\} \text{ and } p \in (0,1].$$

In words, we communicate the coordinates in the support of the coordinator point x_k, together with some coordinates outside the support, randomly selected with some *exploration probability* p.

This adaptive sampling often shows tremendous gains in practice compared to uniform sampling; however, it may not converge in some situations. This is due to two technical points:

(1) The sampling is *not i.i.d.* anymore since the probabilities depend on the points generated by the algorithm.

(2) A *good conditioning* is necessary to allow for a small $p = \mathrm{p}_{\min}$ (with $\mathrm{p}_{\max} = 1$). More precisely, from (8.11), we get that the minimal conditioning to allow for a probability p of selection outside the support is:

$$\kappa_{(\mathcal{P}'_{\mathrm{B-8}})} > \kappa_{\min} := \frac{1-\sqrt{p}}{1+\sqrt{p}}. \tag{8.12}$$

Since we aim at taking p small to communicate little, this is a stringent condition.[63]

[63] This difficulty could be managed with a small algorithmic fix that unfortunately degrades practical performances and a refined analysis, but the upcoming methods directly address this point.

While these two issues appear separate, they can both be overcome by iteratively reconditioning the problem, as developed next.

8.4.3 Proximal reconditioning for adaptive sparsification

Proximal reconditioning methods consist in iteratively regularizing the problem at hand with the squared distance to some center point. We call *outer iteration* the process of (approximatively) solving such a reconditioned problem. At outer loop m, we define worker i's regularized function h_m^i as

$$h_m^i(x) = f^i(x) + \frac{\rho}{2}\|x - x_m\|_2^2 \quad \text{for any } x \in \mathbb{R}^n$$

where ρ is the regularization factor and x_m the center point at outer loop m. The *reconditioned problem for loop m* then writes

$$\min_{x\in\mathbb{R}^n} \; H_m(x) := \sum_{i=1}^{M} \alpha^i \underbrace{\left(f^i(x) + \frac{\rho}{2}\|x - x_m\|_2^2\right)}_{h_m^i(x)} + g(x). \tag{$\mathcal{R}_m$}$$

For μ-strongly convex L-smooth functions (f^i), the regularized functions h^i_m are $(\mu+\rho)$-strongly convex and $(L+\rho)$-smooth. Hence, the condition number of the smooth part of $(\mathcal{R}_m)$ writes

$$\kappa_{(\mathcal{R}_m)} = \frac{\mu+\rho}{L+\rho} \quad \left(\geq \kappa_{(\mathcal{P}'_{\text{B-8}})} = \frac{\mu}{L} \right).$$

The optimal solution of $(\mathcal{R}_m)$ is exactly the proximal point of F/ρ at x_m. This lead to a (inexact) proximal algorithm for solving $(\mathcal{P}'_{\text{B-8}})$, which writes[64]

$$x_{m+1} \approx \operatorname{argmin}_{x\in\mathbb{R}^n} H_m(x) = \operatorname{argmin}_{x\in\mathbb{R}^n} \underbrace{\sum_{i=1}^M \alpha^i f^i(x) + g(x)}_{=F(x)} + \frac{\rho}{2}\|x - x_m\|_2^2 \quad (8.13)$$
$$= \operatorname{prox}_{F/\rho}(x_m).$$

[64] The proximal point algorithm is a standard regularization approach in optimization: it was presented in (Bellman et al., 1966, Chap. 5) to recondition a convex quadratic objective; and popularized by the seminal works (Martinet, 1970; Rockafellar, 1976). The study of the algorithm and its inexact variant, has attracted a lot of attention; see *e.g.* (Fuentes et al., 2012; Güler, 1992; Lin et al., 2017, 2019; Solodov and Svaiter, 2000).

Implementing this algorithm requires an inner algorithm to compute the proximal point (we will use DAve-SPG here) and a rule to stop this algorithm (we will use the standard criteria of (Rockafellar, 1976)).

At the outer iteration m, we run DAve-SPG for solving $(\mathcal{R}_m)$ with i.i.d. non-uniform sparsification probabilities given, for a fixed $0 < c \leq n$, by

$$p_{j,m} = \begin{cases} p_m := \min\left(\dfrac{c}{|\operatorname{null}(x_m)|}; 1\right) & \text{if } x^{[j]}{}_m = 0 \\ 1 & \text{if } x^{[j]}{}_m \neq 0 \end{cases} \quad \text{for all } j \in \{1,..,n\}. \quad (8.14)$$

The sparsification level over outer iterations is then bounded from below by

$$p := \frac{c}{n} \leq \inf_m p_m$$

We now choose the reconditioning parameter ρ from p so that DAve-SPG converges linearly to the solution of the reconditioned problem $(\mathcal{R}_m)$. We know, from (8.11), that this is the case as soon as

$$\kappa_{(\mathcal{R}_m)} = \frac{\mu+\rho}{L+\rho} > \kappa_{\min} \iff \rho > \frac{\kappa_{\min} L - \mu}{1-\kappa_{\min}} \quad \text{with } \kappa_{\min} = \frac{1-\sqrt{p}}{1+\sqrt{p}} \text{ as in (8.12).}$$

To properly handle the strict inequality above, we propose to choose a conditioning which guarantees a $(1-\beta)$ rate for DAve-SPG on the reconditioned problems uniformly over m. Mathematically, for $0 < \beta < p$ (for instance $\beta = p/2$), we choose

$$\rho = \frac{\kappa_{(\mathcal{R}_m)} L - \mu}{1-\kappa_{(\mathcal{R}_m)}} \quad \text{with} \quad \kappa_{(\mathcal{R}_m)} = \frac{1-\sqrt{p-\beta}}{1+\sqrt{p-\beta}}.$$

Then, the contraction factor of DAve-SPG with the maximal stepsize (see (8.9)) for the reconditioned problem $(\mathcal{R}_m)$ becomes

$$\left(\left(\frac{1-\kappa_{(\mathcal{R}_m)}}{1+\kappa_{(\mathcal{R}_m)}}\right)^2 + 1 - p_m\right) = (\pi - \beta + 1 - p_m) = \underbrace{(1-\beta-(p_m-p))}_{=:(1-\beta_m)} \leq 1-\beta < 1.$$

Hence, DAve-SPG converges linearly on the reconditioned problem ($\mathcal{R}_m$). It can thus be safely used as an inner method in the inexact proximal algorithm (8.13) to solve the original problem ($\mathcal{P}'_{\text{B-8}}$).

The remaining part is to the choice of a stopping criterion for the inner loop. We propose to use three different criteria: epoch budget, absolute accuracy, and relative accuracy (called C^1, C^2, and C^3 respectively). Stopping criteria based on accuracy are usually more stringent to enforce (see *e.g.* (Lin et al., 2017, Sec. 2.3) and references therein), however they may bring significant performance improvement when the instantaneous rate is better than the theoretical one.

The resulting algorithm, called Reco.-DAve-SPG, is presented as Algorithm 8.5. Under any of the three stopping criteria, we recover the same convergence result, formalized below.

Algorithm 8.5 Reco.-DAve-SPG on $((\alpha^i), (f^i), g)$

1: Initialize x_0, $n \geq c > 0$, and $\delta \in (0, 1)$.

$$\text{Set } \rho = \frac{\kappa L - \mu}{1 - \kappa} \text{ and } \gamma \in \left(0, \frac{2}{\mu + L + 2\rho}\right] \text{ with } \kappa = \frac{1 - \sqrt{p - \beta}}{1 + \sqrt{p - \beta}}; p = \frac{c}{n} \text{ and } \beta = \frac{c}{2n}.$$

2: **while** the desired accuracy is not achieved **do**

3: Observe the support of x_m, compute p_m as

$$p_{j,m} = \begin{cases} p_m := \min\left(\frac{c}{|\operatorname{null}(x_m)|}; 1\right) & \text{if } x^{[j]}{}_m = 0 \\ 1 & \text{if } x^{[j]}{}_m \neq 0 \end{cases} \qquad \text{for all } j \in \{1, .., n\}.$$

4: Compute an approximate solution of the reconditioned problem ($\mathcal{R}_m$) using DAve-SPG on $\left((\alpha^i), (h^i_m), g; \mathsf{p}_m\right)$ with x_m as initial point and with the stopping criterion:

C^1 (epoch budget): Run DAve-SPG with the maximal stepsize for

$$\mathsf{M}_m = \left\lceil \frac{(1+\delta)\log(m)}{\log\left(\frac{1}{1-\beta+p-p_m}\right)} + \frac{\log\left(\frac{2\mu+\rho}{(1-\delta)\rho}\right)}{\log\left(\frac{1}{1-\beta+p-p_m}\right)} \right\rceil \text{ epochs.}$$

or C^2 (absolute accuracy): Run DAve-SPG until it finds x_{m+1} such that

$$\|x_{m+1} - \operatorname{prox}_{F/\rho}(x_m)\|^2 \leq \frac{(1-\delta)\rho}{(2\mu+\rho)m^{1+\delta}} \|x_m - \operatorname{prox}_{F/\rho}(x_m)\|^2.$$

or C^3 (relative accuracy): Run DAve-SPG until it finds x_{m+1} such that

$$\|x_{m+1} - \operatorname{prox}_{F/\rho}(x_m)\|^2 \leq \frac{\rho}{4(2\mu+\rho)m^{2+2\delta}} \|x_{m+1} - x_m\|^2.$$

5: **end while**

Theorem 8.11. *Let Assumption 8.6 hold. Then, the sequence generated by Reco.-DAve-SPG on $((\alpha^i), (f^i), g)$ with stopping criterion C^1, C^2, or C^3 converges almost surely to a*

minimizer of F and [65]

$$\mathbb{E}\left[\left\|x_{m+1}-x^\star\right\|^2\right] = \tilde{O}\left(\left(1-\frac{\mu}{\mu+\rho/2}\right)^m\right) \quad \textit{for criterion } \mathrm{C}^1;$$

$$\left\|x_{m+1}-x^\star\right\|^2 = \tilde{O}\left(\left(1-\frac{\mu}{\mu+\rho/2}\right)^m\right) \quad \textit{for criteria } \mathrm{C}^2, \mathrm{C}^3.$$

[65] *We use the standard notation: $a_m = \tilde{O}((1-r)^m)$ denotes that there exists C, p such that $a_m \leq Cm^p(1-r)^m$.*

Proof. The proofs of the three cases follow the standard rationale of the seminal work (Rockafellar, 1976) and the recent (Lin et al., 2017). They are detailed in (Grishchenko et al., 2021). □

This result thus establishes that Reco.-DAve-SPG converges linearly to a solution of ($\mathcal{P}'_{\mathrm{B}-8}$). This means that Reco.-DAve-SPG has qualitatively the same behavior as DAve-PG, with the additional feature of having sparse local updates and therefore sparse worker-to-coordinator communications. In other words, our algorithm is similar to the baseline in terms of iterations, but it is expected to be faster in terms of communications (more precisely in terms of quantity of information exchanged between coordinator and workers) which would result in a wallclock gain in practice, as shown in Section 8.5. Before this, we further investigate in the next section the theoretical gain of our sparsification technique in the case of sparse optimal solutions.

Remark 8.12 (Acceleration). In this paper, we are interested in sparsifying communications and we primarily consider the reconditioning aspect of proximal methods, leaving aside other aspects, including acceleration. Indeed the iterations of the inexact proximal algorithm can be accelerated using Nesterov's method (Güler, 1992; Nesterov, 1983) and the recent works (Lin et al., 2017, 2019) also propose accelerated and quasi-Newton variants of the inexact proximal algorithm as a meta-algorithm to improve the convergence of optimization methods driven by machine learning applications. The developments of this section could be extended to accelerated versions, following the meta-algorithm of (Lin et al., 2017). ◀

8.4.4 Identification and consequences

The iterates of proximal algorithms usually *identify* the optimal structure as covered in Chapter 3. In the case of ℓ_1-regularization, this means that proximal algorithms produce iterates that eventually have the same support as the optimal solution of ($\mathcal{P}'_{\mathrm{B}-8}$). In the previous sections, we showed that our sparsified method converges after proximal reconditioning. By construction, the coordinator-to-worker communications depend on the structure of x_k (the coordinator point of the inner method), which is the output of a proximal operator on g. We can thus show that x_k eventually become sparse after some iterations. This automatically makes our algorithm a "two-way sparse" algorithm.

Now, we make the additional assumption that our problem has a strongly sparse solution. This assumption is divided into two parts: i) the regularizer g should induce a *stable* support at the optimum (through its proximity operator, in the sense of (PQC – Proximal Gradient) in Chapter 4); and ii) this optimal support $\mathrm{supp}(x^\star)$ should be small with respect to the ambient dimension.

Assumption 8.13 (Strongly sparse optimal solution)**.** Assumption 8.6 holds and the unique solution $x^\star$ of Problem ($\mathcal{P}'_{\mathrm{B}-8}$) verifies

i) there is $\varepsilon > 0$ such that

$$\text{for all } y \in \mathcal{B}\left(x^\star - \sum_{i=1}^M \alpha^i \nabla f^i(x^\star), \varepsilon\right), \quad \operatorname{supp}(x^\star) = \operatorname{supp}\left(\operatorname{prox}_g(y)\right);$$

ii) the size $s^\star = |\operatorname{supp}(x^\star)|$ of the optimal support is small compared to n: $s^\star \ll n$.

The proximal qualification condition in i) can be made explicit when $g = \lambda\|\cdot\|_1$:

$$\sum_{i=1}^M \alpha^i \nabla^{[j]} f^i(x^\star) \in (-\lambda, \lambda) \quad \text{for all } j \in \operatorname{null}(x^\star).$$

This condition matches the nondegeneracy condition for sparse solutions commonly admitted for exact recovery in machine learning; see *e.g.* (Nutini et al., 2019; Sun et al., 2019). The interest of the general assumption i) is that it accounts for a variety of sparsity-inducing regularizations, including weighted ℓ_1-norms, "group" ℓ_1/ℓ_q-norms; see (Bach et al., 2012, Sec. 3.3).

Using this assumption and combining the arguments of the proofs of Theorem 8.8 and Theorem 8.11, we can show that the (inner and outer) iterates identify the optimal support in finite time with probability one.

Theorem 8.14 (Identification)**.** *Let Assumption 8.13 hold. Then, the* outer and inner *iterates of* Reco.-DAve-SPG *identify the optimal structure in finite time: with probability one, there is $\Lambda < \infty$ such that*

$$\operatorname{supp}(x_{m,k}) = \operatorname{supp}(x_m) = \operatorname{supp}(x^\star) \quad \textit{for any } k \textit{ and all } m \geq \Lambda$$

where $x_{m,k}$ denotes the k-th iterate produced by DAve-SPG *during the m-th outer loop.*

This identification has two consequences on communications in our distributed setting. First, identification implies that the variables communicated by the coordinator to the workers will eventually be sparse. Second, this sparsity is also leveraged in the sparsification strategies of Reco.-DAve-SPG where only the coordinates in (x_m) are randomly zeroed. Thus, for sparsity inducing problems, our distributed algorithm has, structurally, *two-way sparse adaptive communications*.

Even better, once this identification occurs, the rate of the inner algorithm DAve-SPG dramatically improves to match the rate of its non-sparsified version DAve-PG. To get this improved rate, a small additional property is needed on the regularizer: g has to be separable with respect to $\operatorname{supp}(x^\star)$ *i.e.* $g(x) = g_1([x]^{\operatorname{supp}(x^\star)}) + g_2([x]^{\operatorname{null}(x^\star)})$ which holds true for almost all sparsity inducing regularizations (Bach et al., 2012, Sec. 3.3).

Theorem 8.15 (Improved rate)**.** *Let Assumption 8.13 hold. Then, the* inner *iterates of* Reco.-DAve-SPG *benefit from an improved rate after identification. There is $\Lambda < \infty$ such that for all $m > \Lambda$ and $k \in [k_\ell, k_{\ell+1})$, using the maximal stepsize $\gamma = \frac{2}{\mu+L+2\rho}$,*

$$\left\|x_{m,k} - {x^\star}_m\right\|^2 \leq \left(\frac{1-\kappa_{(\mathcal{R}_m)}}{1+\kappa_{(\mathcal{R}_m)}}\right)^{2m} \left\|x_m - {x^\star}_m\right\|^2$$

where $x_{m,k}$ denotes the k-th iterate produced by DAve-SPG *during the m-th outer loop.*

Proof. The proof follows similar arguments as the one of Theorem 5.17 in Chapter 5. Since $x_{m,k}$ has the same support as $x^\star$ after some time and g is separable with respect to the optimal support, the other coordinates do not play a role anymore. This is true for the main variable x but also for the average of the agents contributions $\overline{x}$. This means that DAve-SPG eventually generates the same iterates as DAve-PG in the support of the optimal solution while the coordinates outside the support are null. Hence, they have the same rate. A detailed proof is provided in (Grishchenko et al., 2021). □

Theorem 8.15 intuitively tells us that after identification, the obtained iterates *no longer depend on the sparsification* since the coordinates in the support are always selected with Reco.-DAve-SPG (which is the fundamental reason behind our sparsification choice (8.14)). This means that the rate of our method is eventually the same as if no sparsification was made (*i.e.* when $p_{j,m} = 1$ for all j, m). Since our sparsified method sends less coordinates per iteration, it will thus outperform its non-sparsified counterpart in terms of communications.

8.4.5 Communication complexity

Finally, we study in this section the asymptotic communication complexity of our method in terms of *number of coordinates (*real *numbers) exchanged between the coordinator and the workers.*[66] To do so, we define our communication complexity as

$$\mathrm{C}(\varepsilon) = (\mathrm{c}^{c\to w} + \mathrm{c}^{w\to c})K\mathrm{L}\mathrm{M}(\varepsilon)$$

where i) $\mathrm{c}^{c\to w}$ (resp. $\mathrm{c}^{w\to c}$) is the (expected) number of coordinates communicated from the coordinator to the active worker (resp. from the active worker to the coordinator) during one iteration and K is the average number of iterations per epoch; ii) L is the (expected) number of inner epochs per outer loop; and iii) $\mathrm{M}(\varepsilon)$ is the number of outer loops to reach accuracy ε.

Focusing on the final regime of the algorithm when identification has taken place (*i.e.* when $|\operatorname{supp}(x_{m,k})| = |\operatorname{supp}(x^\star)| = s^\star$ as per Theorem 8.14), we get:[67]

	$\mathrm{c}^{c\to w}$	$\mathrm{c}^{w\to c}$	L	$\mathrm{M}(\varepsilon)$
Reco.-DAve-SPG	$s^\star$	$s^\star + c$	$\tilde{O}(1/(\gamma(\mu+\rho)))$	$O((\mu+\rho/2)/\mu \log(1/\epsilon))$
DAve-PG	$s^\star$	n	1	$O(\kappa_{(\mathcal{P}'_{\mathrm{B-8}})} \log(1/\epsilon))$

As a consequence, in terms of communication complexity, our algorithm offers the following gain (ratio of communication complexities: the greater, the better for our method) over DAve-PG, when the parameter c is of the order of $s^\star$ compared to n

$$\tilde{O}\left(\frac{1+\kappa_{(\mathcal{P}'_{\mathrm{B-8}})}}{1-\kappa_{(\mathcal{P}'_{\mathrm{B-8}})}} \min\left\{\sqrt{\frac{c}{s^\star}}; \sqrt{\frac{s^\star}{c}}\right\} \frac{n+s^\star}{\sqrt{ns^\star}}\right).$$

This gain shows a product of three terms. The first one is greater than 1 and depends on the conditioning; the second one is in $(0, 1]$ but should be not far from 1, provided that the final sparsity is not too poorly estimated. Finally, the last term fully exhibits the merits of adaptive sparsification with a term in $n + s^\star$ for DAve-PG which is much greater than the $\sqrt{ns^\star}$ for Reco.-DAve-SPG. This last term thus shows a nice dependence in the dimension of the problem and optimal solution for the proposed method. This comparison is formalized in (Grishchenko et al., 2021) and illustrated numerically in the next section.

[66] Even though sparse vectors have to encode the location of non-zero elements, the communication cost of Reco.-DAve-SPG can narrowly approximated by the number of coordinates sent.

[67] For Reco.-DAve-SPG: following (8.14), we have $\mathrm{L} = \tilde{O}(1/(\gamma(\mu+\rho))$ for both C^2 and C^3 from Theorem 8.15, as well as $\mathrm{M}(\varepsilon) = O\left(\frac{\mu+\rho/2}{\mu} \log\left(\frac{1}{\varepsilon}\right)\right)$ from Theorem 8.11. For DAve-PG: we can consider that every epoch constitutes an outer iteration with $\rho = 0$, hence $\mathrm{L} = 1$ and the outer loop complexity $\mathrm{M}(\varepsilon)$ boils down to the epoch complexity.

8.5 Numerical Illustrations

In this section, we run some numerical experiments to illustrate the behavior of the different methods presented in this chapter. To do so, we consider the problem of minimizing a logistic loss with elastic-net regularization on a dataset split among M workers. The objective can be written as

$$\min_{x\in\mathbb{R}^n} \frac{1}{M}\sum_{i=1}^{M} \underbrace{\sum_{j\in\mathcal{S}_m{}^i} \log\left(1+\exp(-b_j \mathrm{a}_j^\top x)\right) + \frac{\lambda_2}{2}\|x\|_2^2}_{f^i(x)} + \underbrace{\lambda_1\|x\|_1}_{g(x)},$$

where for each example j, the pair (a_j, b_j) represents the features $\mathrm{a}_j \in \mathbb{R}^n$ together with the corresponding label $b_j \in \{-1, 1\}$. The set $\mathcal{S}_m{}^i$ corresponds to the examples stored locally at machine i; the total number of examples is denoted by m.

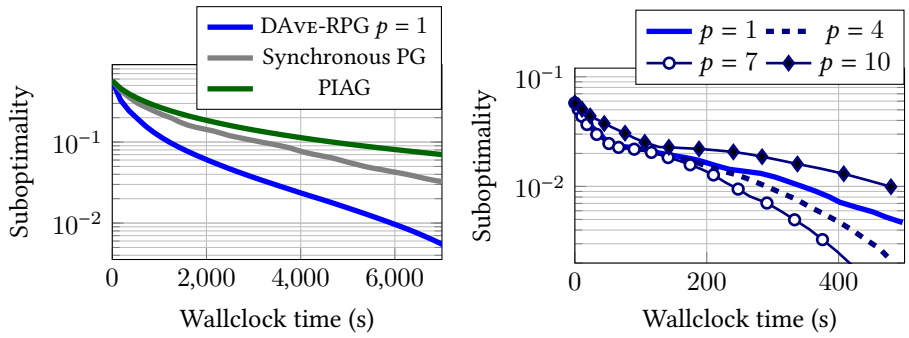
The experiments were run on a CPU cluster, one core corresponding to one worker. Each core had 4 GB of memory and used one thread to produce updates. The code was written in Python using standard libraries only. The datasets used for the experiments are URL ($n = 3,231,961$, $m = 2,396,130$), KDDA ($n = 20,216,830$, $m = 8,407,752$), madelon ($n = 500$, $m = 2,000$), and RCV1 train ($n = 47,236$, $m = 20,242$) from the LIBSVM datasets library (Chang and Lin, 2011).

8.5.1 Performance of DAve-RPG

In Fig. 8.5a, we plot the suboptimality versus wallclock time for the proposed DAve-RPG with $p = 1$, the usual synchronous proximal gradient, and PIAG for the KDDA dataset with $\lambda_1 = 10^{-6}$ and $\lambda_2 = 1/m$. For all algorithms, we used the maximal stepsize. We use the first $200,000$ features and split evenly the examples over 60 workers. Even in this case where the workers have similar computational loads, the performance of DAve-RPG is clearly better than that of the synchronous gradient descent. DAve-RPG also outperforms PIAG, notably thanks to its robustness (as expected from Fig. 8.3). Then, in Fig. 8.5b, we illustrate the repetition of local iterations: we plot the suboptimality versus wallclock time for the proposed DAve-RPG with $p = 1, 4, 7, 10$ on the full URL dataset with $\lambda_1 = 10^{-6}$ and $\lambda_2 = 1/m$ split evenly over 100 workers. We see that a tradeoff appears between computation and communications/updates; in this particular case, the performance improves up to $p = 7$ and then degrades afterwards.

8.5.2 Delay tolerance

In Fig. 8.6, we exhibit the resilience of our algorithm to delays by introducing additional simulated delays. We use the RCV1 train dataset distributed evenly among $M = 10$ machines, meaning that a long delays for one machine would hold out 10% of the data. Delays are simulated by randomly stopping any machine for some random time. We can see that while delays obviously affect the convergence rate, the speed remains comparable. This is an important feature of our algorithm, especially when looking at the maximal delay $d_k = \max_i d_k^i$ record which is varying a lot as expected from a practical point of view. Notice that the delays are only upper bounded by a large value $d \approx 300$ which would deeply affect the stepsize and convergence of competitor algorithms, but not ours.



(a) KDDA (b) URL

Figure 8.5: Illustration of the wallclock time performance of DAve-RPG.

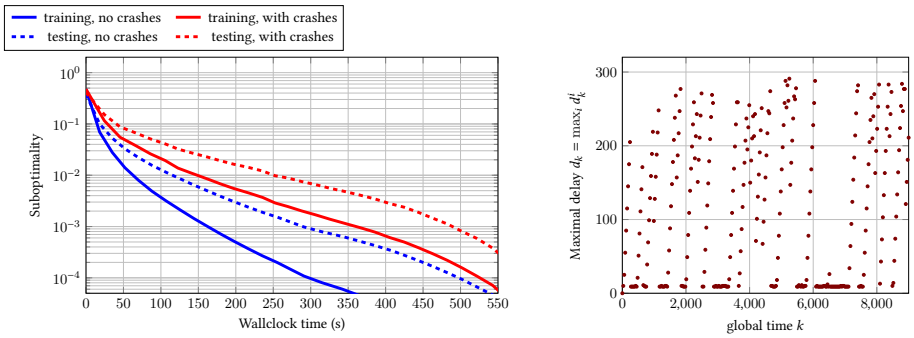


Figure 8.6: Illustration of the resilience of DAve-RPG to additional delays. On the left, performance on RCV1. On the right, maximal delay over one run.

8.5.3 *Effect of adaptive sparsification*

In this section, to have very sparse solutions, we take hyperparameters as follows: for madelon, $\lambda_2 = 0.001$ and $\lambda_1 = 0.03$ chosen to reach 99% sparsity (*i.e.* $|\operatorname{supp}(x^\star)| = 5$); for RCV1, $\lambda_2 = 0.0001$ and $\lambda_1 = 0.001$ to reach 99.7% sparsity (*i.e.* $|\operatorname{supp}(x^\star)| = 61$). The datasets are split evenly between the M workers ($M = 10$ for madelon and $M = 20$ for rcv1).

We illustrate our sparsified algorithm Reco.-DAve-SPG (Algorithm 8.5) for different the amount of randomly chosen coordinates c. We take a simplified stopping criteria C^1 with $\mathsf{M}_\ell = 1$; we thus stop the inner iterations after two passes over the data, following the practical guidelines of Catalyst (Lin et al., 2017). We observe that this simple stopping rule gives similar empirical convergence as C^3 with respect to both iterations and scalars exchanged (without the additional computational cost of the test, for a numerical illustration, see the supplement of (Grishchenko et al., 2021)).

We display the performances of the algorithms in three ways:

- size of support vs number[68] of inner iterations, showing the identification
- functional suboptimality vs total number of inner iterations,
- functional suboptimality vs communication cost, modelled as the number of couples (coordinate, value) sent from and to the coordinator.

[68] Identification is illustrated by plotting the size of $\operatorname{supp}(x_k)$. For Reco.-DAve-SPG, x_k refers to the value of x after k exchanges with the master in total.

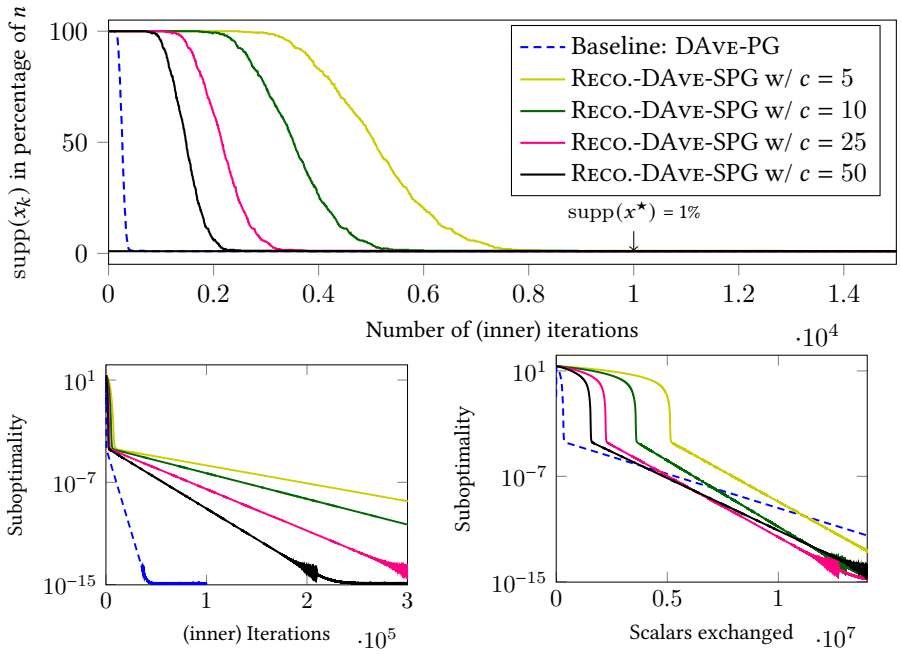


Figure 8.7: Comparison on the madelon logistic regression problem.

We make the following observations from Fig. 8.7. When the support of iterates is far from the optimal one, sparsification is generally bad for convergence in terms of iterations (as shown by the slopes in the plots "suboptimality vs iterations"), and even in terms of communications (see the beginning of the curves "suboptimality vs exchanges"). But, when the iterates get closer to the optimal support, adaptive sparsification becomes highly beneficial as illustrated by the slopes of the plots "suboptimality vs exchanges".

Since there is no guarantee that the currently identified support is the optimal one, it is impossible to restrict ourselves to a subset of the coordinates; here comes the need for our adaptively sparsified method, that keeps exploring dimensions, additionally to those in the current support. The number of randomly chosen coordinates c has an impact: we see on that (relatively) small and large values of c (yellow and black curves) lead to slightly worse slopes on the convergence plots, compared to c being in the range of 1 to 3 times the optimal support (green and pink curves) which is our recommendation both theoretically and in practice.

Finally, in order to mitigate the above-mentioned negative effects of sparsification in the first iterations for large problems, we propose to use a warmstart strategy: in the first iterations, we use a non-sparsified (typically DAve-PG) to allow for a sharp initial functional decrease, leading to some partial identification; after this warmstart we switch to our sparsified method to fully benefit from identification. This strategy is illustrated in Fig. 8.8, with warmstarted algorithms on the right-hand-side vs. the non-warmstarted ones on the left-hand-side. We see a drastic improvement in terms of communication offered by the quick identification, for all versions of the sparsified method.

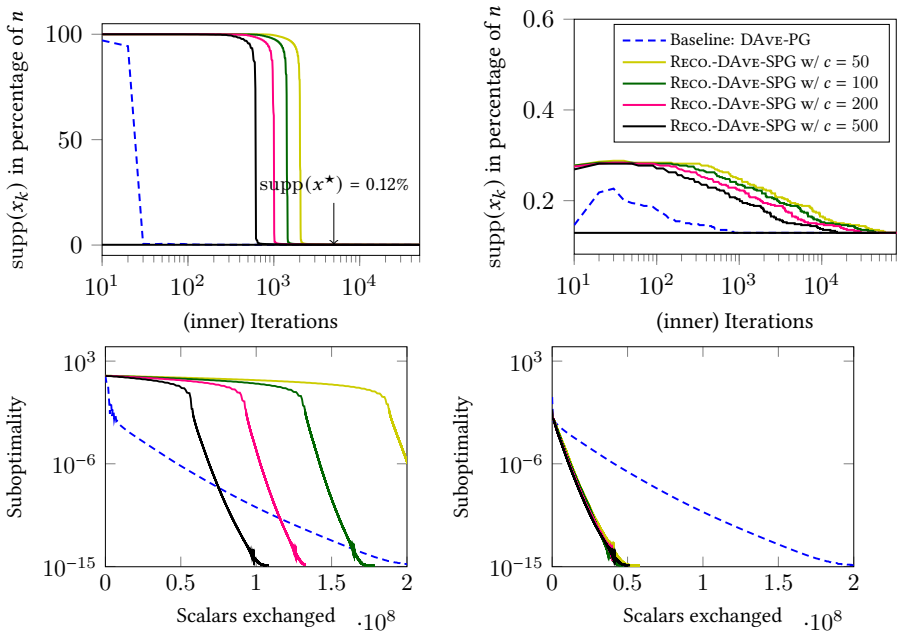


Figure 8.8: Comparison on the RCV1 logistic regression problem. On the right-hand side plots, the algorithms are warm-started to an initial suboptimality of 10^{-2} and density of 1%, reached within less than 5% of the total number of exchanges to target precision.

8.6 Concluding remarks

In this chapter, we described original ways of extending the proximal gradient algorithm for asynchronous distributed optimization. A key property of these algorithms is that they do not require unrealistic assumptions nor any kind of bound on delays.

A key feature of these methods is the conservation by the coordinator of the average of all the agents updates with the right weighting. This lead us to two key theoretical findings: i) an epoch-based analysis adapted to any kind of delays; and ii) the use of the same stepsizes as in the classical proximal gradient algorithm.

This construction is highly flexible and notably enables the use of local tailored stepsizes, local iterations for each worker, sparsification, etc.

Furthermore, when the nonsmooth part of the objective is sparsity-inducing, we showed that we could recondition our methods to adaptively sparsify the points exchanged between the workers and the coordinator leading to a nice improvement of the communication complexity of the method.

DISCUSSION AND CONCLUDING REMARKS

9 Perspectives

Et dans mes petits poings sanglants
d'où pendaient quatre ailes dorées,
je haussais vers le ciel
la gloire de mon père
en face du soleil couchant.
Marcel Pagnol – *La gloire de mon père (1957)*

I am often lead to new research topics by stumbling upon beautiful mathematical objects, reasoning, or computational frameworks. As an applied mathematician, I like problems that mix interesting developments with numerical observations, or try to make the most of a computing framework's abilities while preserving theoretical guarantees. A particularity that appears in several of my works is the wish to have adaptive/automatic methods in the sense that if my goal is to exploit some particular information, it should not be a manually tuned parameter of the algorithm. For instance, if we want to exploit some sparsity structure, the pattern should be learned by the method. If we want to adapt to asynchronous oracles or varying number of agents, we should not rely on a bound on the response time; or if such a bound is necessary, the algorithm should explicitly handle the cases where it is violated. A drawback of this approach is that the performance gain of these adaptive methods is often hard to evaluate (at least mathematically), in the gray zone between "blind" and "omniscient" methods. That is why i) many theoretical results are of the form "we are at least as good as the blind method but can be as good as the omniscient one if everything goes right"; and ii) a numerical implementation is necessary to evaluate the proposed method, confront our intuition to practice, and foster new ideas.

Going further with nonsmooth structure

The numerical exploitation of the nonsmooth structure in optimization problems is probably my favorite research topic at the moment. As developed in Part A, I particularly like how the structure can be automatically identified by a proximity operator, partially at first, then exactly for qualified solutions. This found a practical echo in the case of regularized data science problems where structure is important since it bears valuable information. The objectives of these problems as a sum of a smooth plus a nonsmooth structure-enhancing term naturally leads us to higher-order methods on the identified structure manifold. Numerically, this is especially interesting when the dimension of this manifold is small compared to the ambient space.

Exploring this idea in Chapter 6, a necessary prerequisite (and maybe shortcoming) was the need for i) an explicit proximity operator (giving the structure of the output), and ii) a full Riemannian "panoply" (in particular, a second-order retraction

and derivative). These requirements are often verified for in data science problems, which were our primary target then, but also open on to interesting leads.

Beyond explicit structure

First, we can observe that some nonsmooth functions may not have an explicit proximity operator while still exhibiting exploitable structure. In particular, this can be the case for compositions of functions in the form $g \circ \phi$ where ϕ is a smooth mapping and g has an explicit proximal operator with an associated structure (as in Chapter 3). Such an objective is the typical target of prox-linear methods (Bolte et al., 2020; Cartis et al., 2011; Drusvyatskiy and Paquette, 2019; Lewis and Wright, 2016) which are also known to identify some structure (see *e.g.* Section 4.5 in (Lewis and Wright, 2016)). However, the quantity that identifies lies in the input space of g and not the variable space. Mathematically, a prox-linear operation finds a direction d_{k+1} such that $y_{k+1} := \phi(x_k) + \nabla\phi(x_k)d_{k+1}$ has some structure induced by g; however, it is impossible in general to recover a point x_{k+1} such that $\phi(x_{k+1}) = y_{k+1}$ and thus all structure is lost when restoring the feasibility of the main iterate. In addition, prox-linear steps are not always easily computable even when the proximity operator of the outer function is explicit. Nevertheless, the case of compositions is very attractive in practice and also allows to handle the additive case considered in Part A by a direct reformulation, which opens the way to more generic implementations.

More generally, the interesting situation that appears above is the following: some nearby structure is identified, and even though we cannot project efficiently on it, we have an explicit local equation for this manifold. Furthermore, we known that this manifold bears all the non-differentiability points of the function. Hence, if we manage to exploit this information numerically, we could outperform generic nonsmooth optimization methods.

A typical example of this case is the minimization of the maximal eigenvalue of an affine combination of symmetric matrices: $\min_{x\in\mathbb{R}^n} \lambda_{\max}(A_0 + \sum_{i=1}^n x^{[i]}A_i)$ (see *e.g.* equation (8.7) in (Lewis and Wylie, 2019)). The problem is nonsmooth but the maximal eigenvalue of matrix obtained with the solution $A^\star := A_0 + \sum_{i=1}^n x^{\star[i]}A_i$ is often of multiplicity $r > 1$. Then, the manifold $\mathcal{M} = \{x \in \mathbb{R}^n : \lambda_{\max}(A_0 + \sum_{i=1}^n x^{[i]}A_i)$ is of multiplicity $r\}$ contains all non-differentiability points locally around the solution in $\mathbb{R}^n$ and the objective is smooth on $\mathcal{M}$. Even though it is hard to project on $\mathcal{M}$, this information can be used. For instance, (Noll and Apkarian, 2005) directly solve the problem $\min_{x\in\mathcal{M}} \lambda_{\max}(A_0 + \sum_{i=1}^n x^{[i]}A_i)$ by an algorithm close to Sequential Quadratic Programming thus deferring the nonsmoothness to the constraints and taking full advantage of second-order models, see also (Oustry, 1999, 2000; Overton, 1988; Shapiro and Fan, 1995). This resulted in practical successes but relying on the knowledge of the final multiplicity or on heuristics.

An interesting project would thus be to see if this kind of structure could be identified by using more explicitly the properties of the outer function g (typically, the proximity operator of $\lambda_{\max}$ can serve as a structure oracle giving a guess on the final multiplicity). This kind of approach can be placed between the full-knowledge case, where we know exactly the structure manifold, and structure adaptive methods (such as the k-Bundle of (Lewis and Wylie, 2019)), where the structure is approximated. Indeed, we would exploit here the proximity operator of the nonsmooth function to obtain an educated guess of the local structure.

We noted in preliminary experiments that this nearby structure information can also be used to check optimality numerically. Indeed, it seems possible in many

cases of interest to extend continuously the subdifferential of g around the identified manifold. Doing so (and under some additional qualification), we can form a quantity that converges to 0 as the iterates approach a critical point (of $g \circ \phi$) situated on the identified manifold, even though the function is nonsmooth (and even sharp) at this point (which is problematic for termination in general, see (Shamir, 2020) for a recent discussion). This could provide a more explicit alternative to the optimality measures based on bundle information (Lewis and Wylie, 2019).

Finally, it seems important for such a project to be backed by an efficient implementation. Fortunately, the implementations of manifold representations, routines, and optimization have been constantly evolving, from Matlab toolboxes as Manopt (Boumal et al., 2014) to Python and Julia implementations (Axen et al., 2021; Bergmann, 2019; Townsend et al., 2016) that offer remarkable performances as well as support for advanced computing techniques such as automatic differentiation (*e.g.* Tensorflow (Abadi et al., 2016) or PyTorch (Paszke et al., 2017) in Python, the various packages in Julia (Bezanson et al., 2017) –ForwardDiff, BackwardDiff, Zygote,...–, JAX (Bradbury et al., 2018), etc.) which is ubiquitous in machine learning (see (Bolte and Pauwels, 2020) for theoretical aspects). This would allow us to clearly see where this additional information leads to a performance gain compared to nonsmooth quasi-Newton methods (Keskar and Wächter, 2019; Lewis and Overton, 2013), gradient sampling (Burke et al., 2020), or manifold sampling (Larson et al., 2016, 2020) in a slightly different context. Additionally, this is a motivation to be more "generic" in terms of input by requiring only the functions' implementation, leaving the differentiation to automatic differentiation routines (as mentioned above) or constructing it by querying function values at several points as in (Hare et al., 2020) in our context.

Structure stability: Optimization, Statistics, & Implementation

A longer-term perspective for my ressearch would be to provide "structure stability" guarantees given a smooth objective and a nonsmooth regularization. Typically, we are given by a black-box method some ε-optimal and structured point, and we want to have an indication if this structure is stable (*e.g.* if diminishing/tilting the objective would not necessarily destroy the structure).

A typical use case is when a (sub-)problem involves an external routine (*e.g.* a QP with sparsity inducing constraints) and post-processing the solution (*e.g.* manually zeroing small coordinates). The use of the function implementation and proximity operator of the regularization may enable us to output some stability or qualification guarantee for the candidate structure, which can be quite useful for practitioners when structure bears valuable information such as feature importance.

This would also be interesting in the stochastic case where the part of the function is only accessible though a noisy oracle. In this case, the structure stability problem could be posed as a distributionally robust objective; see *e.g.* (Blanchet et al., 2018; Esfahani and Kuhn, 2018; Rahimian and Mehrotra, 2019).

Going one step further, I would like to understand more precisely the relation between this kind of structure bringing a numerically exploitable conditioning gain and properties like RIP for low-rank matrix recovery (see *e.g.* (Charisopoulos et al., 2021; Ding et al., 2020) for recent takes on the topics). More generally, I wish to understand better the connection between "optimal" sparsity as presented in this manuscript and "statistical" sparsity, especially in terms of local qualification and rate with respect to the recovery possibilities. This is particularly interesting in view of the link between computational complexity and recovery or estimation developed for

instance in (Bandeira et al., 2018; Donoho and Tsaig, 2008; Roulet et al., 2020).

Another angle of attack would be to build on the idea of performance estimation by semi-definite programming introduced in (Drori and Teboulle, 2014) in order to study the "worst identification" performance for classes of functions and algorithms. For instance, we could formulate the problem of minimizing the structure of the final point (*e.g.* by maximizing the 0 semi-norm for sparsity patterns) under the constraint that it was generated by a prescribed number of proximal gradient steps using the tools developed in (Taylor et al., 2017a, 2018) (see also (Taylor et al., 2017b) for an implementation).

Beyond minimization but still with structure

Variational inequalities have recently attracted a considerable amount of attention in machine learning as a flexible paradigm for "optimization beyond minimization". In these problems, finding an optimal solution does not necessarily involve minimizing a loss function but rather optimizing against a "worst-case", an adversary, or some uncertainty element (adversarial machine learning, robust reinforcement learning, learning in games, etc.).

Given a $\mathbb{R}^n \to \mathbb{R}^n$ Lipchitz vector field v and a closed convex constraint set $\mathcal{X} \subset \mathbb{R}^n$, solving the associated (Stampacchia) variational inequality consists in finding $x^\star$ such that

$$\langle v(x^\star), x - x^\star \rangle \geq 0 \text{ for all } x \in \mathcal{X}.$$

For instance, finding a saddle-point of a smooth convex-concave objective $(x_1, x_2) \mapsto \Phi(x_1, x_2)$ can be directly formulated as a variational inequality with the vector field $v = (\nabla_{x_1}\Phi, -\nabla_{x_2}\Phi)$. A direct extension of projected gradient descent fails to converge in general but a this can be remediated by resorting to the extragradient method of (Korpelevich, 1976) which can be written as

$$\left\{ \begin{array}{rcl} x_{k+1/2} = & \mathrm{proj}_{\mathcal{X}}\left(x_k - \gamma_k\, v(x_k)\right) \\ x_{k+1} = & \mathrm{proj}_{\mathcal{X}}\left(x_k - \gamma_k\, v(x_{k+1/2})\right) \end{array} \right. .$$

Now, we may also extend this method to Bregman geometries. For a proper strictly convex lower semicontinuous function h and a continuous selection ∇h of its subdifferential, the associated Bregman divergence between $u \in \mathrm{dom}\,\partial h$ and $x \in \mathrm{dom}\, h$ is $D^h(u, x) = h(u) - h(x) - \langle \nabla h(x), u - x \rangle$. The projection can be replaced using the Bregman proximal mapping defined for $x \in \mathrm{dom}\,\partial h$ as $P_x(y) = \mathrm{argmin}_{u \in \mathcal{X}} \{\langle y, x - u\rangle + D^h(u, x)\}$, which leads to the mirror prox algorithm (Nemirovski, 2004):

$$\left\{ \begin{array}{rcl} x_{k+1/2} = & P_{x_k}\left(-\gamma_k\, v(x_k)\right) \\ x_{k+1} = & P_{x_k}\left(-\gamma_k\, v(x_{k+1/2})\right) \end{array} \right. .$$

The behavior and "computationability" of the algorithm depends heavily on the choice of h, often called the distance generating function. Taking h in accordance with $\mathcal{X}$ can ease the proximal step by replacing the Euclidean projection (corresponding to $h(x) = x^2/2$) by a simpler update in a different metric. Typically, for the n-dimensional simplex $\mathcal{X} = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x^{[i]} = 1\}$, the proximal mapping associated with the entropy $h(x) = \sum_{i=1}^n x^{[i]} \log(x^{[i]})$ simply consists in a multiplicative update and a rescaling, which is much simpler than the Euclidean projection.

However, this has consequences in terms of identification. If the solution $x^\star$ lies on the boundary of $\mathcal{X}$, the iterates of the algorithm will also lie on the boundary after some time in the Euclidean case (under some qualification condition). In Bregman geometries, this may not be the case since the output of the proximal map is sandwiched between $\operatorname{ri}\mathcal{X}$ and $\mathcal{X}$, depending on the domain of ∂h. Intuitively, if ∇h explodes at the boundary, the iterates can never identify the boundary of the domain near $x^\star$ (since it is at an infinite distance in this geometry). But this identification failure does not mean that the behavior of the algorithm around boundary and interior solution is the same.

Motivated by our recent works on related methods in the stochastic setting (Azizian et al., 2021; Hsieh et al., 2019, 2020a), a current research interest is the precise characterization of the last iterate convergence rate of mirror prox when $x^\star$ is on the boundary. Preliminary results show that the rate depends heavily on the distance generating function used and that it is attained with very different stepsizes. To obtain these results, our approach is to define a *Legendre exponent* that characterizes tightly how the Bregman divergence to the optimum $D^h(x^\star, x)$ behaves compared to the ambient norm locally around a border solution $x^\star$.

These results are local by nature and a practical drawback is that the stepsize policy needed to obtain the best rates depend on the knowledge of the structure of $x^\star$ (which may not be identified). An interesting direction would thus be to try and devise this optimal structure, for instance by performing one Euclidean step from time to time. This would enable us to adapt the geometry as well as the optimal stepsize on the run.

Collaborative optimization with loosely connected agents

Distributed methods are employed in a continuum of situations ranging from i) using computation/storage parallelism on hyper-connected clusters to boost the wallclock computing time; to ii) variable-sized groups of autonomous agents that seldom cooperate to solve global task. These problems are very different due to their various objectives and, most importantly, the communication possibilities, notably in terms of synchronization.

In this spectrum, I am drawn towards the second end of the range, focusing more on loosely connected fleet of autonomous agents. In particular, a direction that I am currently pursuing is the question of collaborative optimization in open networks. In this situations, agents can come and go (even for good) which means that the objective to minimize can change with time. A natural idea is thus to model the situation as an *online* optimization problem. Building on our preliminary work on online optimization with delays (Hsieh et al., 2020b), we can show that dual averaging offers satisfactory properties for this situation thanks to the *equal treatment of gradients* in the updates (Hsieh et al., 2021). This kind of reasoning can offer interesting solution in practice and in theory, that still have to be confronted to more real-life problems.

In the vein of loosely connected networks (more far-fetched and exploratory but also stimulating), is it possible to have a flocking dynamics when solving an optimization (*e.g.* a regression problem)? More precisely, let each agent be given a local function; suppose that at each time, it minimizes it and averages its current parameter with the closest parameters in the networks (*i.e.* it only communicates with agents having a similar parameter). Does this behavior lead to a situation where the agents are clustered and eventually agree on a common parameter per cluster as in the (cluster) flocking dynamics of *e.g.* (Beaver and Malikopoulos, 2021; Biccari et al.,

2019; Blondel et al., 2005; Olfati-Saber, 2006)? In terms of statistical regression, this would mean that the agents somehow clustered themselves by distribution proximity.

A more down-to-earth perspective I find particularly interesting is considering the agents as "social" individuals instead of bare computing units. For instance, this raises the problem of fairness between the agents which could be tackled by replacing the sum of the agents by a median (already studied in the context of Byzantine attacks, see (El-Mhamdi et al., 2021) and references therein) but also with a Conditional Value-at-Risk (Laguel et al., 2020) or a re-weighting procedure (I imagine something in the vein of Iteratively Reweighted Least Squares or Weiszfeld's algorithm, see (Daubechies et al., 2010; Weiszfeld, 1937)). Finally, the question of communication incentives to guide the communication of autonomous agents would also be an original direction.

Another research project that came to my attention while working with inexact bundles (see Section 7.4 and (Iutzeler et al., 2020)) is the question of inexact distributed oracles. Indeed, distributed (splitting) methods are based on combining oracles from several sources. These oracles were exact (sub-)gradients in most of Part B. However, there is a large variety of splitting methods based on other types of objects such as proximity operators (in *e.g.* Chambolle-Pock, Condat-Vũ, 3OP, Davis-Yin) or related subproblems (in ADMM); see (Condat et al., 2019) for a recent overview. These subproblem are usually solved using external solvers (QP or otherwise) that can often be stopped at a prescribed precision. Spending too much time generating a high-precision solution is intuitively nefarious if the disagreement between the agents is large or if the functions evolve over time; nevertheless, this kind of reasoning is difficult to formalize mathematically.

Last, but not least, as many of my colleagues, I am prone to drop what I am doing if presented with a cute little problem.[69]

[69] https://xkcd.com/356/ – Isn't it, Panayotis!

Closing words

Thank you for reading this manuscript. While this text is rather personal by nature, it is the fruit of numerous discussions and collaborations with the persons I have had the chance to meet. To them, I extend my most sincere thanks.

As for the future, I can only hope that the next years will be as stimulating as these last ones. And on these words, back to blackboard!

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

Azary Abboud, Franck Iutzeler, Romain Couillet, Merouane Debbah, and Houria Siguerdidjane. Distributed production-sharing optimization and application to power grid networks. *IEEE transactions on Signal and Information Processing over Networks*, 2(1):16–28, 2015.

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Hedy Attouch and Jérôme Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116 (1):5–16, 2009.

Hedy Attouch and Juan Peypouquet. The rate of convergence of Nesterov's accelerated forward-backward method is actually faster than $1/k^2$. *SIAM Journal on Optimization*, 26(3):1824–1834, 2016.

Jean-François Aujol and Charles Dossal. Stability of over-relaxations for the forward-backward algorithm, application to fista. *SIAM Journal on Optimization*, 25(4): 2408–2433, 2015.

Seth D Axen, Mateusz Baran, Ronny Bergmann, and Krzysztof Rzecki. Manifolds. jl: An extensible julia framework for data analysis on manifolds. *arXiv preprint arXiv:2106.08777*, 2021.

Arda Aytekin, Hamid Reza Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv:1610.05507*, 2016.

Waïss Azizian, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. The last-iterate convergence rate of optimistic mirror descent in stochastic variational inequalities. In *Conference on Learning Theory*, pages 1–31. PMLR, vol. 134, 2021.

Léonard Bacaud, Claude Lemaréchal, Arnaud Renaud, and Claudia Sagastizábal. Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioners. *Computational Optimization and Applications*, 20(3):227–244, 2001.

Francis Bach. Consistency of trace norm minimization. *The Journal of Machine Learning Research*, 9:1019–1048, 2008.

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

Afonso S Bandeira, Amelia Perry, and Alexander S Wein. Notes on computational-to-statistical gaps: predictions using statistical physics. *Portugaliae Mathematica*, 75(2): 159–186, 2018.

Gilles Bareilles and Franck Iutzeler. On the interplay between acceleration and identification for the proximal gradient algorithm. *Computational Optimization and Applications*, 77(2):351–378, 2020.

Gilles Bareilles, Franck Iutzeler, and Jérôme Malick. Newton acceleration on manifolds identified by proximal-gradient methods. *arXiv preprint arXiv:2012.12936*, 2020a.

Gilles Bareilles, Yassine Laguel, Dmitry Grishchenko, Franck Iutzeler, and Jérôme Malick. Randomized progressive hedging methods for multi-stage stochastic programming. *Annals of Operations Research*, 295(2):535–560, 2020b.

Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

Logan E Beaver and Andreas A Malikopoulos. An overview on optimal flocking. *Annual Reviews in Control*, 2021.

Amir Beck. *First-order methods in optimization*. SIAM, 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009a.

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009b.

R.E. Bellman, R.E. Kalaba, and J. Lockett. *Numerical Inversion of the Laplace Transform*, pages 143–144. Elsevier, New York, 1966.

Alessandro Benfenati, Emilie Chouzenoux, and J-C Pesquet. A proximal approach for a class of matrix optimization problems. *arXiv preprint arXiv:1801.07452*, 2018.

R. Bergmann. Manopt.jl, 2019. URL https://www.manoptjl.org.

Nilson Bernardes. On nested sequences of convex sets in Banach spaces. *Journal of Mathematical Analysis and Applications*, 389(1):558 – 561, 2012.

Dimitri Bertsekas. On the goldstein-levitin-polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174–184, 1976.

Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Pascal Bianchi, Walid Hachem, and Iutzeler Franck. A stochastic coordinate descent primal-dual algorithm and applications. In *International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014a.

Pascal Bianchi, Walid Hachem, and Franck Iutzeler. A stochastic primal-dual algorithm for distributed asynchronous composite optimization. In *Global Conference on Signal and Information Processing (GlobalSIP)*, pages 732–736. IEEE, 2014b.

Pascal Bianchi, Walid Hachem, and Franck Iutzeler. A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2016.

Umberto Biccari, Dongnam Ko, and Enrique Zuazua. Dynamics and control for multi-agent networked systems: A finite-difference approach. *Mathematical Models and Methods in Applied Sciences*, 29(04):755–790, 2019.

José Bioucas-Dias and Mário Figueiredo. A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.

Jose Blanchet, Karthyek Murthy, and Fan Zhang. Optimal transport based distributionally robust optimization: Structural properties and iterative schemes. *arXiv preprint arXiv:1810.02403*, 2018.

Vincent D Blondel, Julien M Hendrickx, Alex Olshevsky, and John N Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2996–3000. IEEE, 2005.

Jerome Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. *arXiv preprint arXiv:2006.02080*, 2020.

Jérôme Bolte, Trong Phong Nguyen, Juan Peypouquet, and Bruce W Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, pages 1–37, 2015.

Jérôme Bolte, Zheng Chen, and Edouard Pauwels. The multiproximal linearization method for convex composite problems. *Mathematical Programming*, 182(1):1–36, 2020.

Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007.

J. Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

J. Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

Nicolas Boumal. An introduction to optimization on smooth manifolds. Available online, May 2020. URL http://www.nicolasboumal.net/book.

Nicolas Boumal, Bamdev Mishra, Pierre-Antoine Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(42):1455–1459, 2014. URL https://www.manopt.org.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Olivier Briant, Claude Lemaréchal, Ph Meurdesoif, Sophie Michel, Nancy Perrot, and François Vanderbeck. Comparison of bundle and classical column generation. *Mathematical programming*, 113(2):299–344, 2008.

Ronald E Bruck Jr. On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 61(1):159–164, 1977.

Sergio V. B. Bruno, Leonardo A. M. Moraes, and Welington de Oliveira. Optimization techniques for the Brazilian natural gas network planning problem. *Energy Systems*, 8(1):81–101, Feb 2017.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

James V Burke and Jorge J Moré. On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197–1211, 1988.

James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. In *Numerical Nonsmooth Optimization*, pages 201–225. Springer, 2020.

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted ℓ 1 minimization. *Journal of Fourier analysis and applications*, 14(5-6): 877–905, 2008.

Wenfei Cao, Jian Sun, and Zongben Xu. Fast image deconvolution using closed-form thresholding formulas of lq (q= 1/2, 2/3) regularization. *Journal of Visual Communication and Image Representation*, 24(1):31–41, 2013.

Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal on Optimization*, 21(4):1721–1739, 2011.

Alejandro Catalina, Carlos M Alaíz, and José R Dorronsoro. Revisiting fista for lasso: Acceleration strategies over the regularization path. In *ESANN*, 2018.

Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

Antonin Chambolle and Charles Dossal. On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". *Journal of Optimization theory and Applications*, 166(3):968–982, 2015.

Antonin Chambolle, Ronald A De Vore, Nam-Yong Lee, and Bradley J Lucier. Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing*, 7(3):319–335, 1998.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

Vasileios Charisopoulos, Yudong Chen, Damek Davis, Mateo Díaz, Lijun Ding, and Dmitriy Drusvyatskiy. Low-rank matrix recovery with composite optimization: good conditioning and rapid convergence. *Foundations of Computational Mathematics*, pages 1–89, 2021.

Rick Chartrand and Wotao Yin. Nonconvex sparse regularization and splitting algorithms. In *Splitting methods in communication, imaging, science, and engineering*, pages 237–249. Springer, 2016.

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

Patrick L Combettes and Jean-Christophe Pesquet. Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping. *SIAM Journal on Optimization*, 25(2):1221–1248, 2015.

Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

Laurent Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.

Laurent Condat, Daichi Kitahara, Andrés Contreras, and Akira Hirabayashi. Proximal splitting algorithms: A tour of recent advances, with new twists. *arXiv preprint arXiv:1912.00137*, 2019.

Rafael Correa and Claude Lemaréchal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62(2):261–275, 1993.

Aris Daniilidis, Warren Hare, and Jérôme Malick. Geometrical interpretation of the predictor-corrector type algorithms in structured optimization problems. *Optimization*, 55(5-6):481–503, 2006.

Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *arXiv preprint arXiv:2101.09545*, 2021.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

Welington de Oliveira. Target radius methods for nonsmooth convex optimization. *Operations Research Letters*, 45(6):659 – 664, 2017.

Welington de Oliveira and Claudia Sagastizábal. Level bundle methods for oracles with on-demand accuracy. *Optimization Methods and Software*, 29(6):1180–1209, 2014.

Welington de Oliveira and Mikhail Solodov. Bundle methods for inexact data. tech. report, 2018.

Welington de Oliveira, Claudia Sagastizábal, and Susana Scheimberg. Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization*, 21 (2):517–544, 2011.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems (NeurIPS)*, pages 1646–1654, 2014a.

Aaron Defazio, Justin Domke, and Tiberio Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pages 1125–1133, 2014b.

Ron S Dembo and Trond Steihaug. Truncated-newtono algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26(2):190–212, 1983.

Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical analysis*, 19(2):400–408, 1982.

John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in neural information processing systems (NeurIPS)*, pages 2160–2168, 2011.

Yunzi Ding, Dmitriy Kunisky, Alexander S Wein, and Afonso S Bandeira. The average-case time complexity of certifying the restricted isometry property. *arXiv preprint arXiv:2005.11270*, 2020.

David L Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

David L Donoho and Yaakov Tsaig. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54(11): 4789–4812, 2008.

Asen L Dontchev. Perturbations, approximations and sensitivity analysis of optimal control systems. 1983.

Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1):451–482, 2014.

Dmitriy Drusvyatskiy and C Kempton. Variational analysis of spectral functions simplified. *arXiv preprint arXiv:1506.05170*, 2015.

Dmitriy Drusvyatskiy and Adrian S Lewis. Optimality, identifiability, and sensitivity. *Mathematical Programming*, 147(1-2):467–498, 2014.

Dmitriy Drusvyatskiy and Courtney Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1):503–558, 2019.

Louis Dubost, Robert Gonzalez, and Claude Lemaréchal. A primal-proximal heuristic applied to the french unit-commitment problem. *Mathematical programming*, 104(1): 129–151, 2005.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

Vincent Duval and Gabriel Peyré. Sparse regularization on thin grids i: the lasso. *Inverse Problems*, 33(5):055008, 2017.

El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, and Lê-Nguyên Hoang. On the strategyproofness of the geometric median. *arXiv preprint arXiv:2106.02394*, 2021.

Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1):115–166, 2018.

Jalal Fadili, Jerome Malick, and Gabriel Peyré. Sensitivity analysis for mirror-stratifiable convex functions. *SIAM Journal on Optimization*, 28(4):2975–3000, 2018.

Jalal M Fadili, Guillaume Garrigos, Jérôme Malick, and Gabriel Peyré. Model consistency for learning with mirror-stratifiable regularizers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso. In *International Conference on Machine Learning*, pages 333–342, 2015.

Mário AT Figueiredo and Robert D Nowak. An em algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

Frank Fischer and Christoph Helmberg. A parallel bundle framework for asynchronous subspace optimization of nonsmooth convex functions. *SIAM Journal on Optimization*, 24(2):795–822, 2014.

Antonio Frangioni. Standard bundle methods: untrusted models and duality. In *Numerical Nonsmooth Optimization*, pages 61–116. Springer, 2020.

Antonio Frangioni and Enrico Gorgone. Bundle methods for sum-functions with "easy" components: applications to multicommodity network design. *Mathematical Programming*, 145(1-2):133–161, 2014.

Pierre Frankel, Guillaume Garrigos, and Juan Peypouquet. Splitting methods with variable metric for kurdyka–łojasiewicz functions and general convergence rates. *Journal of Optimization Theory and Applications*, 165(3):874–900, 2015.

Marc Fuentes, Jérôme Malick, and Claude Lemaréchal. Descentwise inexact proximal algorithms for smooth optimization. *Computational Optimization and Applications*, 53(3):755–769, 2012.

Guillaume Garrigos, Lorenzo Rosasco, and Silvia Villa. Thresholding gradient methods in hilbert spaces: support identification and linear convergence. *ESAIM: Control, Optimisation and Calculus of Variations*, 26:28, 2020.

Arthur M Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

Pontus Giselsson and Stephen Boyd. Monotonicity and restart in fast gradient methods. In *53rd IEEE Conference on Decision and Control*, pages 5058–5063. IEEE, 2014.

Tobias Glasmachers and Urun Dogan. Accelerated coordinate descent with adaptive coordinate frequencies. In *Asian Conference on Machine Learning*, pages 72–86, 2013.

Dmitry Grishchenko, Franck Iutzeler, and Jérôme Malick. Proximal gradient methods with adaptive subspace sampling. *Mathematics of Operations Research*, 2020.

Dmitry Grishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. Distributed learning with sparse communications by identification. *SIAM Journal on Mathematics of Data Science*, 3(2):715–735, 2021.

Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

Hubert Halkin. Implicit functions and optimization problems without continuous differentiability of the data. *SIAM Journal on Control*, 12(2):229–236, 1974.

Keith B Hall, Scott Gilpin, and Gideon Mann. Mapreduce/bigtable for distributed optimization. 2010.

Robert Hannah and Wotao Yin. On unbounded delays in asynchronous parallel fixed-point algorithms. *arXiv:1609.04746*, 2016.

Robert Hannah and Wotao Yin. More iterations per second, same quality–why asynchronous algorithms may drastically outperform traditional ones. *arXiv:1708.05136*, 2017.

Filip Hanzely, Konstantin Mishchenko, and Peter Richtarik. Sega: Variance reduction via gradient sketching. In *Advances in neural information processing systems (NeurIPS)*, pages 2083–2094, 2018.

Warren Hare and Adrian S Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

Warren Hare and Claudia Sagastizábal. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116(1-2):221–258, 2009.

Warren Hare, Chayne Planiden, and Claudia Sagastizábal. A derivative-free $\mathcal{VU}$-algorithm for convex finite-max problems. *Optimization Methods and Software*, 35 (3):521–559, 2020.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, 1993. Two volumes.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. In *33rd International Conference on Neural Information Processing Systems (NeurIPS)*, pages 6938–6948, 2019.

Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. Explore aggressively, update conservatively: Stochastic extragradient methods with variable stepsize scaling. In *34th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 16223–16234, 2020a.

Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. Multi-agent online optimization with delays: Asynchronicity, adaptivity, and optimism. *arXiv preprint arXiv:2012.11579*, 2020b.

Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. Optimization in open networks via dual averaging. *arXiv preprint arXiv:2105.13348*, 2021.

Wen Huang, Pierre-Antoine Absil, and Kyle A Gallivan. A riemannian bfgs method without differentiated retraction for nonconvex optimization problems. *SIAM Journal on Optimization*, 28(1):470–495, 2018.

Naoki Ito, Akiko Takeda, and Kim-Chuan Toh. A unified formulation and fast accelerated proximal gradient method for classification. *The Journal of Machine Learning Research*, 18(1):510–558, 2017.

Franck Iutzeler. *Distributed Estimation and Optimization in Asynchronous Networks*. PhD thesis, Telecom ParisTech, 2013.

Franck Iutzeler. Distributed computation of quantiles via admm. *IEEE Signal Processing Letters*, 24(5):619–623, 2017.

Franck Iutzeler and Philippe Ciblat. Fully distributed signal detection: Application to cognitive radio. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE, 2013.

Franck Iutzeler and Laurent Condat. Distributed projection on the simplex and ℓ_1 ball via admm and gossip. *IEEE Signal Processing Letters*, 25(11):1650–1654, 2018.

Franck Iutzeler and Julien M Hendrickx. A generic online acceleration scheme for optimization algorithms via relaxation and inertia. *Optimization Methods and Software*, 34(2):383–405, 2019.

Franck Iutzeler and Jérôme Malick. On the proximal gradient algorithm with alternated inertia. *Journal of Optimization Theory and Applications*, 176(3):688–710, 2018.

Franck Iutzeler, Philippe Ciblat, and Jérémie Jakubowicz. Analysis of max-consensus algorithms in wireless channels. *IEEE Transactions on Signal Processing*, 60(11): 6103–6107, 2012.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd Annual Conference on Decision and Control (CDC)*, pages 3671–3676. IEEE, 2013a.

Franck Iutzeler, Philippe Ciblat, and Walid Hachem. Analysis of sum-weight-like algorithms for averaging in wireless sensor networks. *IEEE Transactions on Signal Processing*, 61(11):2802–2814, 2013b.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2015.

Franck Iutzeler, Jérôme Malick, and Welington de Oliveira. Asynchronous level bundle methods. *Mathematical Programming*, pages 319–348, 2020.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems (NeurIPS)*, pages 315–323, 2013.

Bikash Joshi, Franck Iutzeler, and Massih-Reza Amini. Asynchronous distributed matrix factorization with similar user and item based regularization. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 75–78, 2016.

Bikash Joshi, Franck Iutzeler, and Massih-Reza Amini. Large-scale asynchronous distributed learning based on parameter exchanges. *International Journal of Data Science and Analytics*, 5(4):223–232, 2018.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

N Keskar and Andreas Wächter. A limited-memory quasi-newton algorithm for bound-constrained non-smooth optimization. *Optimization Methods and Software*, 34(1): 150–171, 2019.

Kibaek Kim, Cosmin Petra, and Victor Zavala. An asynchronous bundle-trust-region method for dual decomposition of stochastic mixed-integer programming. *SIAM Journal on Optimization*, 29(1):318–342, 2019.

Krzysztof C. Kiwiel. Proximal level bubdle methods for convex nondiferentiable optimization, saddle-point problems and variational inequalities. *Mathematical Programming*, 69(1):89–109, 1995. ISSN 0025-5610.

Krzysztof C Kiwiel. *Methods of descent for nondifferentiable optimization*, volume 1133. Springer, 2006.

Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv:1610.02527*, 2016.

Galina M. Korpelevich. The extragradient method for finding saddle points and other problems. *Èkonom. i Mat. Metody*, 12:747–756, 1976.

Yassine Laguel, Krishna Pillutla, Jérôme Malick, and Zaid Harchaoui. Device heterogeneity in federated learning: A superquantile approach. *arXiv preprint arXiv:2002.11223*, 2020.

Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Manifold sampling for \ell_1 nonconvex optimization. *SIAM Journal on Optimization*, 26(4):2540–2563, 2016.

Jeffrey Larson, Matt Menickelly, and Baoyu Zhou. Manifold sampling for optimizing nonsmooth nonconvex compositions. *arXiv preprint arXiv:2011.01283*, 2020.

Ching-pei Lee. On the iterate convergence and manifold identification of inexact proximal-newton-type methods under a sharpness condition. *arXiv preprint arXiv:2012.02522*, 2020.

Claude Lemaréchal. An extension of davidon methods to nondifferentiable problems. *Mathematical programming study*, 3:95–109, 1975.

Claude Lemaréchal. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer, 2001.

Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254):10, 2012.

Claude Lemaréchal, Arkadi Nemirovskii, and Yurii Nesterov. New variants of bundle methods. *Math. Program.*, 69(1):111–147, 1995.

Claude Lemaréchal, François Oustry, and Claudia Sagastizábal. The u-lagrangian of a convex function. *Transactions of the American mathematical Society*, 352(2):711–729, 2000.

Adrian S Lewis. Nonsmooth analysis of eigenvalues. *Mathematical Programming*, 84 (1):1–24, 1999.

Adrian S Lewis. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13(3):702–725, 2002.

Adrian S Lewis and Jingwei Liang. Partial smoothness and constant rank. *arXiv preprint arXiv:1807.03134*, 2018.

Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

Adrian S Lewis and Stephen J Wright. A proximal method for composite minimization. *Mathematical Programming*, 158(1):501–546, 2016.

Adrian S. Lewis and Calvin Wylie. A simple newton method for local nonsmooth optimization. *arXiv preprint arXiv:1907.11742*, 2019.

Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in neural information processing systems (NeurIPS)*, pages 379–387, 2015.

Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Activity identification and local linear convergence of forward–backward-type methods. *SIAM Journal on Optimization*, 27(1):408–437, 2017a.

Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Local convergence properties of douglas–rachford and alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 172(3):874–913, 2017b.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *The Journal of Machine Learning Research*, 18(1):7854–7907, 2017.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. An inexact variable metric proximal point algorithm for generic quasi-newton acceleration. *SIAM Journal on Optimization*, 29(2):1408–1443, 2019.

Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

Dirk Lorenz and Thomas Pock. An inertial forward-backward algorithm for monotone inclusions. *Journal of Mathematical Imaging and Vision*, 51(2):311–325, 2014.

Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Adaptive coordinate descent. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 885–892. ACM, 2011.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtarik, and Martin Takac. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, pages 1973–1982, 2015.

Paul-Emile Maingé. Convergence theorems for inertial km-type algorithms. *Journal of Computational and Applied Mathematics*, 219(1):223–236, 2008.

Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Jérôme Malick, Welington de Oliveira, and Sofia Zaourar. Uncontrolled inexact information within bundle methods. *EURO Journal on Computational Optimization*, 5(1): 5–29, 2017.

Yura Malitsky and Thomas Pock. A first-order primal-dual algorithm with linesearch. *SIAM Journal on Optimization*, 28(1):411–432, 2018.

Bernard Martinet. Régularisation d'inéquations variationelles par approximation successives. *Revue Française d'Informatique et de Recherche Opérationelle*, R3:154–158, 1970.

Robert Mifflin and Claudia Sagastizábal. A $\mathcal{VU}$-algorithm for convex minimization. *Mathematical programming*, 104(2-3):583–608, 2005.

Scott A Miller and Jérôme Malick. Newton methods for nonsmooth convex minimization: connections among-lagrangian, riemannian newton and sqp methods. *Mathematical programming*, 104(2-3):609–633, 2005.

Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning*, pages 3584–3592, 2018.

Konstantin Mishchenko, Franck Iutzeler, and Jérôme Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *SIAM Journal on Optimization*, 30(1): 933–959, 2020.

Aryan Mokhtari, Mert Gurbuzbalaban, and Alejandro Ribeiro. Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate. *SIAM Journal on Optimization*, 28(2):1420–1447, 2018.

Boris S Mordukhovich. Sensitivity analysis in nonsmooth optimization. *Theoretical aspects of industrial design*, 58:32–46, 1992.

Boris S Mordukhovich. *Variational analysis and generalized differentiation I: Basic theory*, volume 330. Springer Science & Business Media, 2006.

Jean-Jacques Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 255: 2897–2899, 1962.

Jean-Jacques Moreau. Proximité et dualité dans un espace Hilbertien. *Bull. Soc. Math. France*, 93(2):273–299, 1965.

Hongseok Namkoong, Aman Sinha, Steve Yadlowsky, and John C Duchi. Adaptive sampling probabilities for non-smooth optimization. In *International Conference on Machine Learning*, pages 2574–2583, 2017.

Ion Necoara and Andrei Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

Arkadi Semen Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Trong Phong NGuyen. *Kurdyka-Lojaeiwicz and convexity: algorithms and applications*. PhD thesis, Toulouse University, 2017.

Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *arXiv preprint arXiv:1106.5730*, 2011.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Dominikus Noll and Pierre Apkarian. Spectral bundle methods for non-convex maximum eigenvalue functions: second-order methods. *Mathematical Programming*, 104 (2):729–747, 2005.

Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641, 2015.

Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv:1712.08859*, 2017.

Julie Nutini, Mark Schmidt, and Warren Hare. "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letters*, 13(4):645–655, 2019.

Kohei Ogawa, Yoshiki Suzuki, and Ichiro Takeuchi. Safe screening of non-support vectors in pathwise svm computation. In *International Conference on Machine Learning*, pages 1382–1390, 2013.

Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

Welington de Oliveira and Claudia Sagastizábal. Bundle methods in the xxist century: A bird's-eye view. *Pesquisa Operacional*, 34(3):647–670, 2014.

François Oustry. The $\mathcal{U}$-lagrangian of the maximum eigenvalue function. *SIAM Journal on Optimization*, 9(2):526–549, 1999.

Francois Oustry. A second-order bundle method to minimize the maximum eigenvalue function. *Mathematical Programming*, 89(1):1–33, 2000.

Michael L Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM Journal on Matrix Analysis and Applications*, 9(2):256–268, 1988.

Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

Gregory B Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 72(2):383–390, 1979.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. Faster coordinate descent via adaptive importance sampling. *arXiv preprint arXiv:1703.02518*, 2017.

René Poliquin and R Rockafellar. Prox-regular functions in variational analysis. *Transactions of the American Mathematical Society*, 348(5):1805–1838, 1996.

Boris T Polyak. Introduction to optimization. *Optimization Software, New York*, 1987.

Clarice Poon and Jingwei Liang. Trajectory of alternating direction method of multipliers and adaptive acceleration. In *Advances in neural information processing systems (NeurIPS)*, pages 7355–7363, 2019.

Clarice Poon, Jingwei Liang, and Carola-Bibiane Schönlieb. Local convergence properties of saga/prox-svrg and acceleration. *arXiv:1802.02554*, 2018.

Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.

Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 20–27, 2004.

Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016a.

Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016b.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*. Springer Science & Business Media, 2009.

Vincent Roulet, Nicolas Boumal, and Alexandre d'Aspremont. Computational complexity versus statistical performance on sparse recovery problems. *Information and Inference: A Journal of the IMA*, 9(1):1–32, 2020.

Claudia Sagastizábal. Divide to conquer: decomposition methods for energy optimization. *Mathematical programming*, 134(1):187–222, 2012.

Claudia Sagastizábal. A VU-point of view of nonsmooth optimization. *Proceedings of the International Congress of Mathematicians*, 4:3815–3836, 2018.

Katya Scheinberg, Donald Goldfarb, and Xi Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14(3):389–417, 2014.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Ohad Shamir. Can we find near-approximately-stationary points of nonsmooth nonconvex functions? *arXiv preprint arXiv:2002.11962*, 2020.

Alexander Shapiro and Michael KH Fan. On eigenvalue optimization. *SIAM Journal on Optimization*, 5(3):552–569, 1995.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

Vitold Smulian. On the principle of inclusion in the space of the type (b). *Rec. Math. [Mat. Sbornik] N.S.*, 5(47):317–328, 1939.

Mikhail V Solodov and Benar Fux Svaiter. Error bounds for proximal point subproblems and associated inexact proximal point algorithms. *Mathematical programming*, 88 (2):371–389, 2000.

Gilbert W Stewart. Perturbation theory for the singular value decomposition. Technical report, 1998.

Sebastian U Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *Advances in neural information processing systems (NeurIPS)*, pages 4381–4391, 2017.

Tao Sun, Robert Hannah, and Wotao Yin. Asynchronous coordinate descent under more realistic assumptions. In *Advances in neural information processing systems (NeurIPS)*, pages 6183–6191, 2017.

Yifan Sun, Halyun Jeong, Julie Nutini, and Mark Schmidt. Are we there yet? manifold identification of gradient-related proximal methods. In *22nd International Conference on Artificial Intelligence and Statistics*, pages 1110–1119, 2019.

Adrien B Taylor, Julien M Hendrickx, and François Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, 27(3):1283–1313, 2017a.

Adrien B Taylor, Julien M Hendrickx, and François Glineur. Performance estimation toolbox (pesto): automated worst-case analysis of first-order optimization methods. In *56th Annual Conference on Decision and Control (CDC)*, pages 1278–1283. IEEE, 2017b.

Adrien B Taylor, Julien M Hendrickx, and François Glineur. Exact worst-case convergence rates of the proximal gradient method for composite convex minimization. *Journal of Optimization Theory and Applications*, 178(2):455–476, 2018.

Marc Teboulle. A simplified view of first order methods for optimization. *Mathematical Programming*, pages 1–30, 2018.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

James Townsend, Niklas Koep, and Sebastian Weichwald. PyManopt: a Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL https://www.pymanopt.org.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

John Nikolas Tsitsiklis. *Problems in decentralized decision making and computation*. PhD thesis, Massachusetts Inst of Tech, Lab for Information and Decision Systems, 1984.

Samuel Vaiter, Gabriel Peyré, and Jalal Fadili. Low complexity regularization of linear inverse problems. In *Sampling Theory, a Renaissance*, pages 103–153. Springer, 2015.

Samuel Vaiter, Gabriel Peyré, and Jalal Fadili. Model consistency of partly smooth regularizers. *IEEE Transactions on Information Theory*, 64(3):1725–1737, 2017.

Wim van Ackooij and Welington de Oliveira. Level bundle methods for constrained convex optimization with various oracles. *Computation Optimization and Applications*, 57(3):555–597, 2014.

Wim van Ackooij and Jerome Malick. Decomposition algorithm for large-scale two-stage unit-commitment. *Annals of Operations Research*, 238(1):587–613, 2015.

Nuri Denizcan Vanli, Mert Gurbuzbalaban, and Asu Ozdaglar. A stronger convergence result on the proximal incremental aggregated gradient method. *arXiv:1611.08022*, 2016.

Nuri Denizcan Vanli, Mert Gurbuzbalaban, and Asuman Ozdaglar. Global convergence rate of proximal incremental aggregated gradient methods. *SIAM Journal on Optimization*, 28(2):1282–1300, 2018.

Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

Hermann Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, 1912.

Antonio Frangioni Wim van Ackooij. Incremental bundle methods using upper models. Technical Report 2357, Dipartimento di Informatica, TR . University of Pisa, 2016.

Christian Wolf, Csaba I Fábián, Achim Koberstein, and Leena Suhl. Applying oracles of on-demand accuracy in two-stage stochastic programming. A computational study. *European Journal of Operational Research*, 239(2):437–448, 2014.

Stephen J Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079, 1993.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1): 3–34, 2015.

Zongben Xu, Xiangyu Chang, Fengmin Xu, and Hai Zhang. $l_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1013–1027, 2012.

Kôsaku Yosida. *Functional analysis*. springer, 1988.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *International Conference on Machine Learning*, pages 1–9, 2015.

Peng Zhao and Bin Yu. On model selection consistency of Lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

APPENDICES

APPENDIX A Summary of my previous contributions

I trained as engineer before pursuing with a PhD in signal processing around problems related to estimation in wireless sensor networks (Iutzeler, 2013).[70] This rather transverse topic allowed me to develop a taste for different disciplines: signal processing, statistics and probabilities, computer science (especially graph theory), and finally optimization. It is this last theme that has been at the heart of my research since then.

[70] More precisely, I started with the analysis of a rumor spreading method on graphs based on broadcast communications (Iutzeler et al., 2012). Then, I studied the convergence rate of an averaging methods based on broadcast communication (Iutzeler et al., 2013b). Finally, we analyzed the rate of a distributed ADMM method on graphs (Iutzeler et al., 2015).

Decentralized optimization

The questions that interested me most at the end of my PhD and at the beginning of my career were mainly centered on the development of randomized methods for decentralized optimization. Decentralized optimization consists in minimizing the individual functions of the agents of a network under the constraint that the solution found by two neighboring agents must be equal. This leads to the search for a consensus around the minimum of the sum of the functions of the agents when the network is (strongly) connected. To solve this kind of problems, it is natural to separate the problem between the individual minimization on the agents' side and the constraints of equality over the network. This can be directly obtained by splitting methods such as the proximal gradient or the Alternating Direction Method of Multipliers (ADMM). These algorithms can be studied under the formalism of monotone operators, *i.e.* by formulating them as fixed point iterations of a certain operator. Driven by the observation that the coordinates of this operator generated local consensus in the graph, one of the main results of my early career was to show that it was possible to draw some of these coordinates randomly, update them, and leave the others unchanged (Bianchi et al., 2016; Iutzeler et al., 2013a). This kind of result was rather original at the time (2013-2015), and has since been improved in many ways by Patrick Combettes, Jean-Christophe Pesquet, and Wotao Yin for example (Combettes and Pesquet, 2015; Peng et al., 2016).

The above result provides, after careful but straightforward computations, "asynchronous" optimization algorithms in networks where each iteration corresponds to drawing a subset of the graph (*e.g.* two neighbors), make them perform a minimization step on their local function and exchange information between them, without the rest of the network working or communicating (Bianchi et al., 2014a,b).

I have also worked on practical applications of these algorithms. For example, in cognitive radio (Iutzeler and Ciblat, 2013), for flow optimization in electrical networks (DC-OPF) with Mérouane Debbah and Romain Couillet (Abboud et al., 2015), or much more recently (2020) in multi-level stochastic programming (Bareilles et al., 2020b),

with Jérôme Malick and a group of students. Another type of application has been the distributed computation of quantiles of agent values (Iutzeler, 2017) and the distributed projection of these values on the simplex, in collaboration with Laurent Condat (Iutzeler and Condat, 2018).

Decentralized optimization is clearly more representative of my earlier works than my current research interests. Nevertheless, it naturally led me to the study of distributed optimization algorithms at large, always with the idea of avoiding as much as possible synchronous communications between the agents. Thus, I became interested in 2017 in distributed optimization algorithms where several agents communicate asynchronously with a common coordinator. At that time, I started to collaborate with Massih Amini, around Bikash Joshi's thesis, on optimization for distributed learning (Joshi et al., 2016, 2018).

Based on this collaboration, I decided to set up a project with Massih Amini and Jérôme Malick to continue in this direction. This project funded the internship of Konstantin Mishchenko (2017) and the thesis of Dmitry Grishchenko (2017-2020). This allowed us to develop methods for asynchronous learning and to reach out to both the learning and mathematical optimization communities which were presented in Part B.

Numerical optimization

Starting in 2015 (*i.e.* when I was in Louvain-la-Neuve), I also started to focus on numerical optimization methods and more specifically on Nesterov-type acceleration methods. My first intuition was to consider a class of algorithms that can be analyzed by the theory of monotone operators (already used above). However, the results on the acceleration of monotone operators are very restrictive and do not allow to observe in theory the gain seen in practice. This is one of the reasons why this kind of acceleration has been limited for a long time to variants of the gradient algorithm. Thus, I was interested in practical methods of acceleration, trying to be as adaptive as possible to the local geometry of the problem. For example, one of my goals was to provide a common acceleration method for strongly convex and simply convex functions (which require different analyses in the accelerated Nesterov gradient).

As a first step, I designed a generic acceleration algorithm that estimates the current speed of the algorithm (seen as a fixed point iteration) and applies to it the acceleration that would be optimal if the associated operator was linear (*i.e.* as if the algorithm was a gradient descent on a quadratic function) (Iutzeler and Hendrickx, 2019). This technique allows for an agnostic acceleration of a large class of algorithms, even with other techniques than inertia. For instance, I have studied the case of alternating inertia (where the inertial step is applied only every other iteration) which allows to have good results in some cases while being generally more stable than inertia. This lead to a first collaboration with Jérôme Malick around the case of the proximal gradient, where we found that it lead to iterations that monotonically decreased the functional value allowing for a finer analysis based on Kurdyka-Lojasiewicz conditions as mentioned in Sections 4.1.3 and 4.1.4.

Then, after numerous discussions with Jérôme Malick, I started a research project on the practical behavior of the accelerated proximal gradient when the solutions of the problem are sparse (see Section 4.2.2). This line of work eventually lead to the results presented in Part A.

APPENDIX B PUBLICATIONS LIST

Most of my publications are available on my webpage.
Bibliometry in July 2021 from Google scholar: 882 citations – h-index=14.

PREPRINTS

P3- G. Bareilles, F. Iutzeler : *Newton acceleration on manifolds identified by proximal-gradient methods*, arXiv:2012.12936, Dec. 2020.

P2- Y.-G. Hsieh, F. Iutzeler, J. Malick, P. Mertikopoulos : *Multi-Agent Online Optimization with Delays: Asynchronicity, Adaptivity, and Optimism*, arXiv:2012.11579, Dec. 2020.

P1- C. Laclau, F. Iutzeler, I. Redko : *Rank-one partitioning: formalization, illustrative examples, and a new cluster enhancing strategy*, arXiv:2009.00365, Sep. 2020.

JOURNAL ARTICLES & NEURIPS/ICML/COLT CONFERENCES

Nota Bene: I chose to include in this section my journal articles as well as my articles published in NeurIPS, ICML, *and* COLT. *I made this choice since these selective conferences (20% acceptance rate) lead to autonomous articles (without associated journal papers) with a depth similar to journal articles; for these reasons, they belong more in that category than with other conferences (IEEE CDC, ICASSP, etc.).*

A22- W. Azizian, F. Iutzeler, J. Malick, and P. Mertikopoulos: *The last-iterate convergence rate of optimistic mirror descent in stochastic variational inequalities* , 34th Annual Conference on Learning Theory (COLT), 2021.

A21- D. Grishchenko, F. Iutzeler, J. Malick, M.-R. Amini: *Distributed Learning with Sparse Communications by Identification*, SIAM Journal on Mathematics of Data Science, vol. 3, no. 2, pp. 715-735, 2021.

A20- F. Iutzeler, J. Malick: *Nonsmoothness in Machine Learning: specific structure, proximal identification, and applications*, Set-Valued and Variational Analysis, vol. 28, no. 4, pp. 661-678, 2020.

A19- Y.-G. Hsieh, F. Iutzeler, J. Malick, P. Mertikopoulos : *Explore Aggressively, Update Conservatively: Stochastic Extragradient Methods with Variable Stepsize Scaling*, Advances in Neural Information Processing Systems 34 (NeurIPS) spotlight, Dec. 2020.

A18- G. Bareilles, Y. Laguel, D. Grishchenko, F. Iutzeler, J. Malick: *Randomized Progressive Hedging methods for Multi-stage Stochastic Programming* , Annals of

Operations Research, vol. 295, no. 2, pp. 535-560, 2020.

A17- G. Bareilles, F. Iutzeler : *On the Interplay between Acceleration and Identification for the Proximal Gradient algorithm*, Computational Optimization and Applications, vol. 77, no. 2, pp. 351–378, 2020.

A16- D. Grishchenko, F. Iutzeler, and J. Malick : *Proximal Gradient Methods with Adaptive Subspace Sampling*, to appear in Mathematics of Operations Research, 2020.

A15- K. Mishchenko, F. Iutzeler, and J. Malick : *A Distributed Flexible Delay-tolerant Proximal Gradient Algorithm*, SIAM Journal on Optimization, vol. 30, no. 1, pp. 933-959, 2020.

A14- Y.-G. Hsieh, F. Iutzeler, J. Malick, and P. Mertikopoulos : *On the convergence of single-call stochastic extra-gradient methods*, Advances in Neural Information Processing Systems 32 (NeurIPS), Dec. 2019.

A13- F. Iutzeler, J. Malick, and W. de Oliveira : *Asynchronous level bundle methods*, Mathematical Programming, vol. 184, pp. 319-348, 2020.

A12- F. Iutzeler and L. Condat : *Distributed Projection on the Simplex and ℓ_1 Ball via ADMM and Gossip*, IEEE Signal Processing Letters, vol. 25, no. 11, pp. 1650-1654, Nov. 2018.

A11- K. Mishchenko, F. Iutzeler, J. Malick, M.-R. Amini : *A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning*, International Conference on Machine Learning (ICML), PMLR 80:3584-3592, Stockholm (Sweden), July 2018.

A10- F. Iutzeler and J. Malick : *On the Proximal Gradient Algorithm with Alternated Inertia*, Journal of Optimization Theory and Applications, vol. 176, no. 3, pp. 688-710, March 2018.

A9- B. Joshi, F. Iutzeler and M.-R. Amini : *Large-scale asynchronous distributed learning based on parameter exchanges*, International Journal of Data Science and Analytics, vol. 5, no. 4, pp. 223-232, June 2018.

A8- F. Iutzeler and J. M. Hendrickx : *A Generic online acceleration scheme for Optimization algorithms via Relaxation and Inertia*, Optimization Methods and Software, vol. 34, no. 2, 2019.

A7- B. Joshi, M.-R. Amini, I. Partalas, F. Iutzeler, Yu. Maximov : *Aggressive Sampling for Multi-class to Binary Reduction with Applications to Text Classification*, Advances in Neural Information Processing Systems 30 (NeurIPS), Dec. 2017.

A6- F. Iutzeler : *Distributed Computation of Quantiles via ADMM*, IEEE Signal Processing Letters, vol. 24, no. 5, pp. 619-623, May 2017.

A5- P. Bianchi, W. Hachem, and F. Iutzeler : *A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Distributed Optimization*, IEEE Transactions on Automatic Control, vol. 61, no. 10, pp. 2947-2957, Oct. 2016.

A4- F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem : *Explicit Convergence Rate of a Distributed Alternating Direction Method of Multiplier*, IEEE Transactions on Automatic Control, vol. 61, no. 4, pp. 892-904, Apr. 2016.

A3- A. Abboud, F. Iutzeler, R. Couillet, M. Debbah, and H. Siguerdidjane : *Distributed Production-Sharing Optimization and Application to Power Grid Networks*, IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 11, pp. 16-28, March 2016.

A2- F. Iutzeler, P. Ciblat, and W. Hachem : *Analysis of Sum-Weight-like algorithms for*

averaging in Wireless Sensor Networks, IEEE Transactions on Signal Processing, vol. 61, no. 11, pp. 2802-2814, June 2013.

A1- F. Iutzeler, P. Ciblat, and J. Jakubowicz : *Analysis of max-consensus algorithms in wireless channels*, IEEE Transactions on Signal Processing, vol. 60, no. 11, pp. 6103-6107, November 2012.

International Conferences

C12- Y.-G. Hsieh, F. Iutzeler, J. Malick, P. Mertikopoulos : *Optimization in Open Networks via Dual Averaging*, to appear at the 60-th IEEE Conference on Decision and Control (CDC), Austin (USA), 2021. (see arXiv:2105.13348)

C11- M. Chastan, A. Lam, F. Iutzeler: *Unsupervised density based machine learning for abnormal leveling signatures detection*, SPIE Advanced Lithography, Online, Feb. 2021.

C10- D. Grishchenko, F. Iutzeler, M.-R. Amini: *Sparse Asynchronous Distributed Learning*, 27-th International Conference on Neural Information Processing (ICONIP), Online, November 2020.

C9- D. Grishchenko, F. Iutzeler, J. Malick: *Distributed First-order Optimization with Tamed Communications*, Signal Processing with Adaptive Sparse Structured Representations (SPARS workshop), Toulouse (France), July 2019.

C8- B. Joshi, F. Iutzeler, M.-R. Amini: *Asynchronous Distributed Matrix Factorization with Similar User and Item Based Regularization*, 10-th ACM Conference on Recommender Systems (RecSys), Boston (USA), Sept. 2016.

C7- F. Iutzeler, P. Bianchi, P. Ciblat and W. Hachem: *Linear Convergence Rate for Distributed Optimization with the Alternating Direction Method of Multipliers*, 53-rd IEEE Conference on Decision and Control (CDC), Los Angeles (USA), December 2014.

C6- P. Bianchi, W. Hachem and F. Iutzeler: *A Stochastic Primal-Dual algorithm for Distributed Asynchronous Composite Optimization*, 2-nd IEEE Global Conference on Signal and Information Processing (GlobalSip), Atlanta (USA), December 2014.

C5- P. Bianchi, W. Hachem, and F. Iutzeler : *A Stochastic Coordinate Descent Primal-Dual Algorithm And Applications*, 24-th IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Reims (France), September 2014.

C4- F. Iutzeler , P. Bianchi, P. Ciblat and W. Hachem: *Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers*, 52-nd IEEE Conference on Decision and Control (CDC), Florence (Italy), December 2013.

C3- F. Iutzeler and P. Ciblat: *Fully-distributed spectrum sensing: application to cognitive radio*, 21-st European Signal Processing Conference (EUSIPCO), Marrakech (Morocco), September 2013.

C2- F. Iutzeler, P. Ciblat, W. Hachem, and J. Jakubowicz : *A new broadcast based averaging algorithm over wireless sensor networks*, 37-th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto (Japan), March 2012.

C1- F. Iutzeler, J. Jakubowicz, W. Hachem and P. Ciblat : *Distributed estimation of the*

maximum value over a wireless sensor network, 45-th Asilomar Conference on Signals, Systems, and Computer, Pacific Grove (USA), November 2011.