

École polytechnique de Louvain

Automated Estimation of Performance of Optimization Methods

Authors: **Alexia DE POORTER, Marie HARTMAN**
Supervisors: **François GLINEUR, Julien HENDRICKX**
Readers: **Yurii NESTEROV, Adrien TAYLOR**
Academic year 2020–2021
Master [120] in Mathematical Engineering

Abstract

This master thesis analyzes and improves the exact worst-case performance of iterative black-box first-order methods for unconstrained optimization of smooth strongly convex functions.

To characterize and compare optimization methods, we evaluate the exact worst-case performance of their solutions based on performance measures using the toolbox PESTO. This toolbox allows resolving instances of the performance estimation problem by resolving a semidefinite program.

In this master thesis, we focus on the gradient method and the fast gradient method developed by Nesterov. Using PESTO, we observe that, unlike the gradient method, the fast gradient method derives little benefits from the strong convexity property. To overcome this, several versions of the fast gradient method specifically designed for smooth strongly convex functions exist but these are non-robust to wrong estimate of the strong convexity parameter. It is therefore necessary to find versions of the fast gradient method that improve the performance of the classical fast gradient method for smooth strongly convex functions but that either do not depend on the strong convexity parameter or are robust to a wrong estimation of this parameter. To this end, the restart has been introduced: applying a restart consists of stopping the algorithm, erasing the memory and taking the last iterate as new initial point.

This master thesis implements three restarting schemes: the fixed restart which consists of applying a restart after a fixed number of iterations, the adaptive restart which consists of applying a restart when a certain restarting condition is satisfied and the soft restart which consists of applying a fraction of restart at each iteration. First, by using an optimal restarting interval, we show that the fixed restart improves the performance of the fast gradient method by 32% to 60% and is robust especially to underestimation of the strong convexity parameter. Then, we show that the adaptive restart does not improve the worst-case performance of the fast gradient method. Finally, we design several strategies of soft restart. One of them improves by 12% to 25% the performance of the fast gradient method with the optimal restarting interval and is also robust to wrong estimations of the strong convexity parameter.

Key Words: Worst-case performance, Optimization, First-order methods, Fixed restart, Adaptive restart, Soft restart, Fast gradient method.

Acknowledgments

First of all, we would like to thank our advisors Julien Hendrickx and François Glineur for their precious support and help throughout this whole year. They helped us to improve and further developing our work by providing us useful advice and giving us many interesting leads and ideas. Their interest and involvement in our topic were very motivating and helped us to produce this final rendering of our work.

Then, we would like to thank Etienne Huens for helping us with the use of the server, Adrien Taylor for helping us with the toolbox PESTO and Carole Lens for the assiduous proofreading of our master thesis.

Finally, we are very grateful to Sébastien Colla for his constant understanding, help and support throughout the realization of our master thesis. Our discussions about our work often helped us to improve our understanding of the subject. We would particularly like to thank him for his assiduous proofreading of our master thesis.

Contents

Abstract	i
Acknowledgments	iii
1 Introduction	1
2 Problem statement	5
2.1 Worst-case performance analysis	5
2.2 Assumptions	6
2.2.1 Definitions	6
2.3 Standard first-order optimization methods	9
2.3.1 Gradient method - GM	10
2.3.2 Fast gradient method - FGM	12
2.4 Conclusion	15
3 Performance estimation problems	17
3.1 Introduction to performance estimation	17
3.2 Smooth strongly convex interpolation	20
3.3 Convex formulation for performance estimation	22
3.4 PESTO toolbox	26
3.5 Conclusion	28
4 Motivation	29
4.1 Worst-case performance and convergence of GM	29
4.2 Worst-case performance and convergence of FGM	31
4.3 Comparison between GM and FGM	32
4.4 Conclusion	33
Appendices	
4.A Worst-case performance and convergence for $\ \nabla f(x_n)\ _2^2$ as performance criterion	34
4.A.1 Worst-case performance and convergence of GM	34
4.A.2 Worst-case performance and convergence of FGM	34
4.A.3 Comparison between GM and FGM	35
5 State-of-the-art	37

5.1 Accelerated first-order methods	37
5.2 Fast gradient methods with fixed restart - FGM-FR	40
5.2.1 Optimal restarting interval k	41
5.2.2 Convergence rate of FGM-FR	42
5.3 Fast gradient methods with adaptive restart - FGM-AR	43
5.4 Conclusion	45
6 Contribution	47
7 Fast gradient method with fixed restart	49
7.1 Method	49
7.2 Worst-case performance analysis	50
7.3 Optimal restarting interval	53
7.3.1 Strategy	54
7.3.1.1 Example	56
7.3.2 Analytical results	58
7.4 Optimal restarting interval formula	60
7.5 Gain	61
7.6 Robustness	64
7.7 Conclusion	66
8 Fast gradient method with adaptive restart	69
8.1 Method	69
8.2 Implementation	70
8.2.1 If condition	70
8.2.2 Strict inequality	73
8.3 Worst-case performance analysis	74
8.4 Inefficiency of the adaptive restart	77
8.5 Conclusion	80
Appendices	
8.A Worst-case performance analysis for the performance criterion $\ \nabla f(x_N)\ _2^2$	82
8.B Inefficiency of the adaptive restart for the performance criterion $\ \nabla f(x_N)\ _2^2$	82
9 Fast gradient method with soft restart	85
9.1 Method	85
9.1.1 First strategy	86
9.1.2 Second strategy	86
9.1.3 Third strategy	87
9.1.4 Comparison between strategies	87
9.2 Worst-case performance analysis	88
9.3 Optimal restarting factor	92
9.3.1 Analytical results	93
9.4 Optimal restarting factor formula	94
9.5 Gain	95
9.6 Robustness	97

9.7 Conclusion	100
10 Conclusion	103
Bibliography	111

1

Introduction

Optimization problems arise in almost all applications across sciences, whether for important power system planning or for delivering packages in shipping companies, airline reservations systems, image processing, etc. Thanks to the growth of computational capabilities, optimization has become a powerful tool for making or even anticipating decisions in diverse areas: medicine (e.g., optimizing the mix of screening and vaccination strategies for the prevention of cervical cancer), manufacturing (e.g., optimizing a company's production plan by maximizing their profit), economy (e.g., prediction of individual behavior by assuming the individuals maximize their utilities), biology (e.g., modelling Darwin's theory of evolution favoring survival of individuals with most beneficial traits), etc.

More formally, an optimization problem searches the *decision variables* x within a set $\mathcal{X}$ which minimize - or maximize - an *objective function* f :

$$\min_{x \in \mathcal{X}} f(x) \quad (\text{Optimization Problem})$$

In daily life, almost every practical application can be modelled using this optimization framework. However, there are relatively few instances that can be actually solved in practice within a reasonable time. Therefore, as there is no strong guarantee for solving (Optimization Problem) in general, optimizers add several assumptions on both $\mathcal{X}$ and f , e.g., differentiability of f . This allows implementing algorithms with working certifications, i.e., guarantees on the solution obtained from specific algorithms. Those assumptions should facilitate the implementation of efficient algorithms for solving (Optimization Problem) while maintaining a sufficiently rich class of problems including as many applications as possible.

In this master thesis, we consider the following assumptions: optimization problems that are continuous, i.e., whose variables take values within a continuum. More precisely, we assume an unconstrained (Optimization Problem), i.e., $\mathcal{X} = \mathbb{R}^d$, and a smooth (strongly) convex objective function f . There exist parameters that can describe the smoothness and the strong convexity of the function f : we call them respectively L and μ and we say that a function is L -smooth μ -strongly convex. Intuitively, the parameter L is a measure of how fast the function can change and a μ -strongly convex function means that there exists a quadratic lower bound on the growth of the function. The smaller the L and/or the higher the μ , the better the function f is conditioned. These notions are properly

defined in Chapter 2.

Moreover, many methods allow solving (Optimization Problem). In this master thesis, we focus on *iterative black-box first-order* optimization methods. First, an *iterative* method proceeds by iterations during which it determines a sequence of refined approximate solutions, called the iterates, (see red points in Figure 1.1) that gradually approach the target solution (see green point in Figure 1.1). Then, a *black-box* method can only evaluate values and some information of the function f at given points, e.g., at a sequence of iterates, without having direct access to the function. Finally, a *first-order* method only exploits information on values and gradient of the function f .

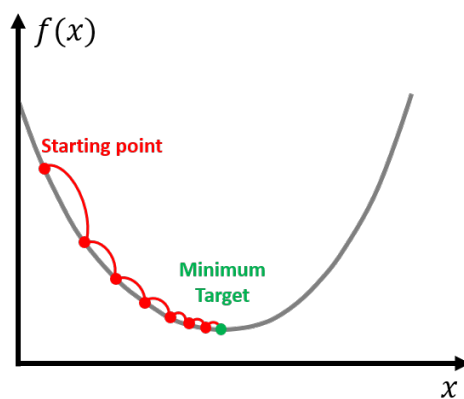


Figure 1.1: Example of an iterative method that reaches the minimum of a function f .

Particularly, we will study two specific iterative black-box first-order optimization methods: the gradient method, noted GM, and the fast gradient method, noted FGM. The gradient method moves the iterates by a fixed step size in the direction of the steepest slope at each iteration: the size of this step often depends on the parameter L . The fast gradient method accelerates the gradient method by using a momentum term. These two methods therefore take into account the parameter L but not the parameter μ . These methods are described more in detail in Chapter 2.

To compare several optimization methods and to determine the most appropriate for solving (Optimization Problem), one can evaluate the performance of these methods. Indeed, we want to obtain guarantees and to characterize how far from the true minimum a solution obtained with a specific method will be after a fixed number of iterations. Thus, guarantees on some performance criteria for specific optimization methods could be obtained.

There are many performance criteria to choose from: the distance between the last iterate and the solution, the difference between the function values at the last iterate and at the solution, etc.

Moreover, several approaches exist to compute the performance. For example, one could look at the performance in the best, the average or the worst case: the best case describes an algorithm behavior under optimal conditions but is rarely considered in practice, the average case allows to be close to the reality but implies complicated statistical calculations and the worst case gives a safe analysis (the worst case is never underestimated) but can

be overly pessimistic. In this master thesis, we focus on worst-case performance, which is the most common choice in the literature. To compute the worst-case performance of first-order algorithms, one approach is to compute theoretical upper bounds on the performance but these bounds are usually not assessed precisely. Then, if a method has a poor performance bound, we cannot conclude that this method does not work well or is less appropriate than another one since this upper bound is not necessarily exact. In this master thesis, to compute the worst-case performance, we use a toolbox called PESTO [THG17a]. This toolbox was recently developed by Adrien Taylor during his thesis [Tay17] and provides an answer to a question that could not be easily answered before:

What is the exact worst-case performance of a given first-order optimization method?

Indeed, the toolbox PESTO allows computing automatically tight worst-case performance for a wide class of iterative first-order optimization methods. Therefore, PESTO is a powerful tool to analyze, compare and choose the appropriate method for solving (Optimization Problem). For example, with this tool, one can analyze the performance of the gradient method and the fast gradient method. More details about the working and the use of this toolbox are explained in Chapter 3.

Ideally, the worst-case performance of an optimization method should be better when the function f that we want to minimize is better conditioned, e.g., when the function f is a L -smooth μ -strongly convex function with a high value of μ or a small value of L . However, in some situations, the fast gradient method derives no or little benefit if the function f turns out to be better conditioned. This will be shown in Chapter 4. To tackle this issue, there exist improved versions of the fast gradient method which take into account these function parameters.

However, these improved methods work very well only when the function parameters are known exactly but in practice, these parameters are generally not known and are difficult to estimate. Then, using wrong estimates of the parameters L and/or μ leads to significant performance losses.

An example of such an improved method is the version of the fast gradient method for smooth strongly convex functions, i.e., a version of the fast gradient method that involves both parameters L and μ in its implementation. This method is denoted as FGM- μ and its implementation is given in Chapter 5.

Then, let us visualize the performance losses of FGM- μ due to wrong estimates of the parameter μ . To this end, we solve the (Optimization Problem) for the class of 1-smooth μ_* -strongly convex functions f and, using the toolbox PESTO, we compare the exact worst-case performance of the classical fast gradient method, i.e., FGM, and its improved version, i.e., FGM- μ , with the exact parameter μ_* and estimates of μ_* : $\mu \in \{\frac{\mu_*}{5}, \frac{\mu_*}{2}, \mu_*, 2\mu_*, 5\mu_*\}$. In other words, we apply FGM- μ on the class of 1-smooth μ_* -strongly convex functions but with wrong estimates of μ_* . We have considered the distance between the last iterate and the solution as performance criterion and the distance between the starting point and the solution is bounded to 1. The results are in Figure 1.2.

As expected, FGM- μ_* significantly improves the worst-case performance of FGM. For example, after 50 iterations, using FGM- μ_* gives a performance that is about 500 times

better than the one obtained with FGM. However, the performance of the improved version FGM- μ suffers a lot from a wrong parameter estimation. For a small number of iterations, it is better to overestimate the μ_* parameter than underestimate it. On the contrary, it is better to underestimate μ_* when the number of iterations increases.

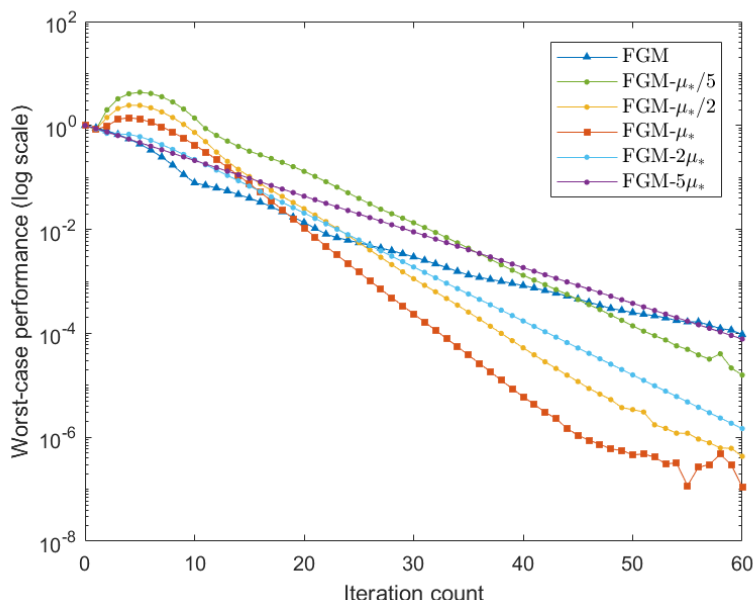


Figure 1.2: Comparison between the performance evolution of FGM and FGM- μ for several values of μ obtained using the toolbox PESTO for the distance between the last iterate and the solution as performance criterion, the distance between the starting point and the solution bounded to 1, $L = 1$ and $\mu_* = 0.05$.

Therefore, the goal of this master thesis will be to derive and study versions of the fast gradient method that does not suffer from these drawbacks. More precisely, we want to determine a version of the fast gradient method which takes advantage of the strong convexity but which does not depend on the strong convexity parameter μ or which is robust even when the estimation of μ is not accurate. To this end, we will use a surprising trick: the restart. The restart was previously introduced and studied in the literature. Applying a restart consists of stopping the algorithm, erasing the memory and taking the last iterate as new initial point. The motivation of this surprising trick will be explained in Chapter 5. Then, several strategies of restart will be implemented and analyzed in Chapter 7, 8 and 9: we will show that some strategies significantly improve the performance of FGM while others give worst performance than FGM.

Let us recall that the goal of this master thesis is to evaluate the performance of methods that solve (Optimization Problem) using some assumptions on the function f . This chapter first defines the problem of finding the worst-case performance. Then, it gives and defines those assumptions and finally describes the methods used in this master thesis along with convergence properties.

2.1 Worst-case performance analysis

To evaluate the performance of a first-order optimization method, one wants to estimate the accuracy of solutions computed by this method. To this end, one usually seeks to analyze the worst-case behavior. More precisely, given a class of problems and a first-order method, the idea is to calculate the worst-case accuracy of solutions for a specific performance criterion that can be obtained after a given number of iterations N . Once identified, this worst-case provides guarantees on the performance of the method. Note that a worst-case performance is exact or tight if there exists a function that achieves the worst-case on the considered performance criterion.

During this thesis, three performance criteria were used to analyze the worst-case performance of the methods, with x_N the last iterate and x_* an optimal solution.

- I The function value accuracy $f(x_N) - f(x_*)$,
- II The distance to the optimal solution $\|x_N - x_*\|_2^2$,
- III The residual gradient norm $\|\nabla f(x_N)\|_2^2$.

The smaller these performance criteria are, the more accurate the method is since the last iterate x_N is closer to the optimal solution x_* when the performance criterion is small.

In the literature, many first-order algorithms for smooth (strongly) convex functions exist and, usually, for these algorithms, lower or upper bounds on the worst-case performance are provided but these bounds are usually not assessed precisely. Therefore, the exact numerical value of the worst-case performance of a specific method is generally unknown.

However, thanks to the PESTO toolbox, computing the exact worst-case performance becomes feasible in some situations. Therefore, in this master thesis, we focus on the

computation of the exact worst-case performance of a given first-order black-box method, on a given class of functions, after a given number of iterations. This will be further explained in Chapter 3.

2.2 Assumptions

In this master thesis, the following assumptions are made to consider only a specific class of problems more easily solvable using specific algorithms.

Firstly, we focus on the class of **first-order optimization algorithms** which only require evaluation of the objective function f and its gradient at a sequence of iterates. In recent years, there has been a resurgence of interest in first-order methods as they proved to be successful to solve very large problem instances, i.e., when the decision space is high-dimensional (see e.g., in [Sun+20; PE10; Yan+19; BB08]). This is due to their low iteration cost since the computational expense of first-order methods depends only mildly on the problem dimension and also due to the development of extremely efficient automatic differentiation software computing gradients automatically.

Secondly, we consider **unconstrained optimization problems** and thus, the decision variables x are within the set $\mathcal{X} = \mathbb{R}^d$. Moreover, we focus on **L -smooth μ -strongly convex optimization problems**, i.e., optimization problems with μ -strongly convex objective function f having a Lipschitz continuous gradient with positive constant L on the convex set $\mathbb{R}^d$. Therefore, in this thesis, the objective function f belongs to the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ of μ -strongly convex and L -smooth functions. Those assumptions allow to have strong theoretical guarantees and, therefore, ensure an adequate behavior of the algorithms on these problems.

2.2.1 Definitions

Let us describe in detail the properties of the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$.

Convexity

I Convex set

From an intuitive perspective, a convex set is a set that should entirely contain the line segment between any two points in that set.

Definition 2.1 (convex set). A set $Q \subseteq \mathbb{R}^d$ is *convex* if for any $x, y \in Q$ and for any $\lambda \in [0, 1]$:

$$\lambda x + (1 - \lambda)y \in Q.$$

In this master thesis, we assume an unconstrained (Optimization Problem) and therefore, our set is equal to $Q = \mathbb{R}^d$ and is convex. Moreover, we focus on non-empty closed convex sets. It is very convenient to work with closed sets because non-closed sets may turn simple problems into ill-defined ones, e.g., when the minimum is not attained.

II Convex function

From an intuitive perspective, a convex function is such that the tangent at any point of the function is always below the functional value, as shown in Figure 2.1a.

Definition 2.2 (convex function). A differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is *convex* if for any $x, y \in \mathbb{R}^d$:

$$f(y) \geq f(x) + \nabla f(x)^T(y - x).$$

The above definition only works for differentiable functions but since this master thesis only focuses on differentiable optimization problems, it is sufficient to consider this less general definition.

Another definition of convexity for twice differentiable functions can be established as follows. Intuitively, this definition means that the curvature of the function must be at least 0.

Definition 2.3 (convex function). A twice differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is *convex* if for all $x \in \mathbb{R}^d$, we have:

$$\nabla^2 f(x) \succeq 0,$$

where $\nabla^2 f(x)$ is the Hessian matrix.

In optimization theory, the minimization of a convex objective function over a convex feasible set ensures that all local optima are global optima. Without convexity, there is a risk to get stuck into a suboptimal solution. Therefore, convex functions give a huge advantage for solving optimization problems as they allow local search algorithms to guarantee global optimal solutions.

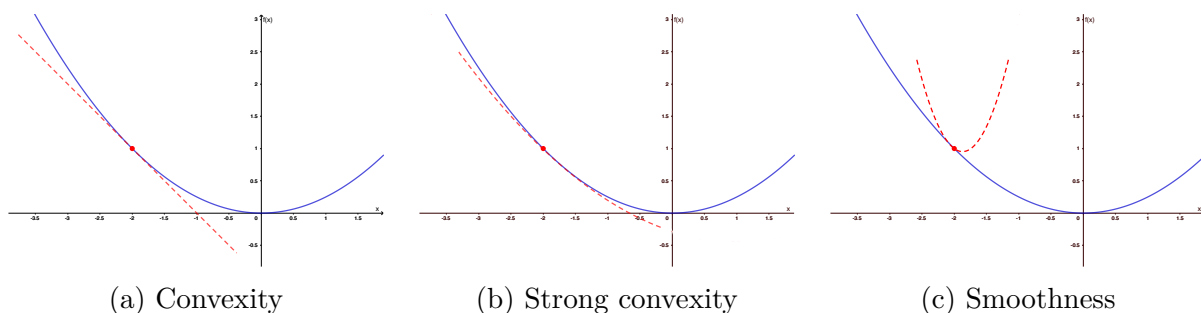


Figure 2.1: Representation of properties of convexity, strong convexity and smoothness.

μ -strong convexity

From an intuitive perspective, a μ -strongly convex function means that there exists a quadratic lower bound on the growth of the function as can be seen in Figure 2.1b.

Definition 2.4 (strongly convex function). A differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is μ -strongly convex with $\mu \geq 0$ if for any $x, y \in \mathbb{R}^d$, we have:

$$f(x) \geq f(y) + \nabla f(y)^T(x - y) + \frac{\mu}{2}\|x - y\|_2^2.$$

As for the convex property, another definition of strong convexity for twice differentiable functions can be established as follows. Intuitively, the following definition means that the curvature of the function must be at least μ .

Definition 2.5 (strongly convex function). A twice differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is μ -strongly convex with $\mu \geq 0$ if for all $x \in \mathbb{R}^d$, we have:

$$\nabla^2 f(x) \succeq \mu I,$$

where $\nabla^2 f(x)$ is the Hessian matrix and I is the identity matrix.

In optimization, strong convexity plays a huge role as it guarantees a linear convergence rate of many gradient descent based algorithms. To be noted that by taking $\mu = 0$ in Definition 2.4 (respectively in Definition 2.5), one comes back to Definition 2.2 (respectively to Definition 2.3), that is the function is convex.

Remark 2.6. By taking $y = x_*$ as any minimizer of f in Definition 2.4, knowing that $\nabla f(x_*) = 0$ for unconstrained minimization problem and using the quadratic lower bound on the function from [Gli19b], the following inequalities can be established for any μ -strongly convex function f :

$$\frac{\mu}{2}\|x - x_*\|_2^2 \leq f(x) - f(x_*) \leq \frac{1}{2\mu}\|\nabla f(x)\|_2^2.$$

***L*-smoothness**

A way of defining smoothness is to require the function to be upper bounded by its first-order development plus a quadratic term as shown in Figure 2.1c. Intuitively, the Lipschitz constant L is a measure of how fast the function can change.

Definition 2.7 (smooth function). A differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -smooth with $L > 0$ if for any $x, y \in \mathbb{R}^d$, we have:

$$f(x) \leq f(y) + \nabla f(y)^T(x - y) + \frac{L}{2}\|x - y\|_2^2.$$

Another definition of smoothness for twice differentiable functions can be established as follows. Intuitively, the following definition means that the curvature of the function must be at most L .

Definition 2.8 (smooth function). A twice differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -smooth with $L > 0$ if for all $x \in \mathbb{R}^d$, we have:

$$\nabla^2 f(x) \preceq LI,$$

where $\nabla^2 f(x)$ is the Hessian matrix and I is the identity matrix.

Remark 2.9. By taking $y = x_*$ as any minimizer of f in Definition 2.7, knowing that $\nabla f(x_*) = 0$ for unconstrained minimization problem and using the quadratic upper bound on the function from [Gli19a], the following inequalities can be established:

$$\frac{1}{2L} \|\nabla f(x)\|_2^2 \leq f(x) - f(x_*) \leq \frac{L}{2} \|x - x_*\|_2^2.$$

Finally, let us properly define the class of functions considered in this master thesis.

Definition 2.10 ($\mathcal{F}_{\mu,L}(\mathbb{R}^d)$). A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ belongs to the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ if f is a L -smooth and μ -strongly convex function with $0 \leq \mu < L < +\infty$.

2.3 Standard first-order optimization methods

We focus on algorithms based on gradient information as it proved to be effective for smooth strongly convex optimization problems, especially the gradient method (GM) and the classical fast gradient method (FGM). To analyze these methods, we can evaluate the convergence rate for some performance criteria.

The rate of convergence is a measure of how fast the difference between the solution and its estimates goes to zero.

Definition 2.11 (convergence rate). A sequence (a_N) that converges to a_* , i.e., such that $\|a_N - a_*\| \rightarrow 0$, is said to have a *rate of convergence* $\nu \in [0, 1]$ if

$$\lim_{N \rightarrow \infty} \frac{\|a_{N+1} - a_*\|}{\|a_N - a_*\|} = \nu.$$

Different values of ν give different rates of convergence:

1. If $\nu = 0$, it is a superlinear rate.
2. If $\nu \in (0, 1)$, it is a linear rate.
3. If $\nu = 1$, it is a sublinear rate.

In order to observe the speed of convergence of the superlinear, linear and sublinear rates, three sequences x_N have been plotted on Figure 2.2. It is easy to deduce that the linear convergence is slower at first but later becomes better than the sublinear convergence after a certain number of iterations. On the semi-logarithmic scale Figure 2.2b, the blue line has indeed a linear convergence rate since it corresponds to a line. Then, we can observe that the green line tends much faster to zero: a sequence that has a superlinear rate converges faster than a sequence that has a linear rate.

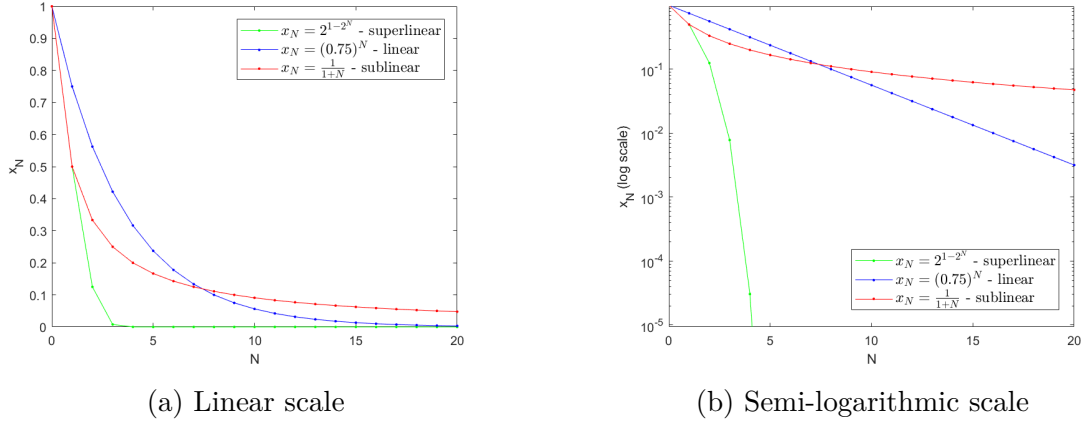


Figure 2.2: Comparison of sequences converging in $\mathcal{O}(c^{\alpha^N})$ with a superlinear rate, in $\mathcal{O}(c^N)$ with a linear rate and in $\mathcal{O}(\frac{1}{N^\beta})$ with a sublinear rate with $c \in (0, 1)$, $\alpha > 1$ and $\beta > 0$.

2.3.1 Gradient method - GM

The gradient method, also called gradient descent, is an iterative optimization algorithm for finding a local minimum of a differentiable function f . At each iteration, the current point x_i is moved in the direction opposite to the gradient of the objective function f at this point, which is the direction of the steepest descent. The method follows the direction of the slope downhill until reaching a local minimum.

The gradient descent algorithm is described in the following and is applied to a function belonging to the class of smooth (strongly) convex functions $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$.

GM

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $x_0 \in \mathbb{R}^d$, $0 \leq h \leq \frac{2}{L}$
 For $i = 0, \dots, N-1$

$$x_{i+1} = x_i - h \nabla f(x_i)$$

Proposition 2.12 (Descent Lemma, [Gli19a]). *Suppose the L -smooth function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ and the initial point $x \in \mathbb{R}^d$. Then if x_+ is given by a gradient step with a fixed step size $0 \leq h \leq \frac{2}{L}$ from x , i.e., $x_+ = x - h \nabla f(x)$, the decrease of the function in x_+ satisfies*

$$f(x) - f(x_+) \geq \frac{h}{2} \|\nabla f(x)\|_2^2.$$

This Descent Lemma guarantees the decrease of the gradient step at each iteration and hence, ensures a convergence to a minimum local. When the function f and the feasible set are convex, all local minima are also global minima and thus, the gradient descent is guaranteed to converge to a global solution.

In what follows, the constant step size h of the gradient method is chosen as $\frac{1}{L}$, i.e.,

inversely proportional to the Lipschitz constant L . Therefore, we study the convergence rates of the gradient method by considering $h = \frac{1}{L}$.

Convergence rates for smooth convex functions

Theorem 2.13 ([DT12, Theorem 3.1]). *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth convex function. Then the iterate x_N of GM with a fixed step size $\frac{1}{L}$ satisfies*

$$f(x_N) - f(x_*) \leq \frac{L}{4N+2} \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution.

Theorem 2.14 ([Nes12]). *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth convex function. Then the iterates x_k for $0 \leq k \leq N$ of GM with a fixed step size $\frac{1}{L}$ satisfies*

$$\min_{0 \leq k \leq N} \|\nabla f(x_k)\|_2^2 \leq \frac{16L^2}{(N^2 + 8N)} \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution.

Theorems 2.13 and 2.14 guarantee that the gradient descent for smooth convex functions converges in $\mathcal{O}(\frac{1}{N})$ for the performance criteria $f(x_N) - f(x_*)$ and in $\mathcal{O}(\frac{1}{N^2})$ for $\min_{0 \leq k \leq N} \|\nabla f(x_k)\|_2^2$. This means that to achieve a performance guarantee of $f(x_N) - f(x_*) \leq \epsilon$, $\mathcal{O}(\frac{1}{\epsilon})$ iterations of the gradient descent must be performed and to achieve a performance guarantee of $\min_{0 \leq k \leq N} \|\nabla f(x_k)\|_2^2 \leq \epsilon$, $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ iterations of the gradient descent must be performed. For these performance criteria, the convergence rate is **sublinear** since $\nu = 1$.

Remark 2.15 ([Nes18, Theorem 2.1.14]). The best guarantee on the performance criterion $\|x_N - x_*\|_2^2$ for smooth convex functions is

$$\|x_N - x_*\|_2^2 \leq \|x_0 - x_*\|_2^2,$$

where x_N is the last iterate of GM with a fixed step size $\frac{1}{L}$ and x_* is any optimal solution.

Indeed, we require that the function is L -smooth convex and, therefore, the worst possible situation is that the worst-case function corresponds to a horizontal line: the gradient of this function is thus null at each iteration and we have $x_{i+1} = x_i$ leading to a stagnation in the gradient descent. Note that Remark 2.15 gives an upper bound on the performance criterion $\|x_N - x_*\|_2^2$ and, therefore, the method could work better in practice.

Thus, for this performance criterion, the convergence rate is also **sublinear** since $\nu = 1$ but there is no guarantee on the number of iterations of the gradient method to be performed in order to reach a performance guarantee of $\|x_N - x_*\|_2^2 \leq \epsilon$.

Convergence rates for smooth strongly convex functions

Theorem 2.16 ([Gli19b]). *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of GM with a fixed step size $\frac{1}{L}$ satisfies*

$$f(x_N) - f(x_*) \leq c^N (f(x_0) - f(x_*)),$$

where x_* is any optimal solution, $0 < c < 1$ and $c = e^{-\frac{1}{e}\frac{\mu}{L}}$.

Theorem 2.17 ([Gli19b]). Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of GM with a fixed step size $\frac{1}{L}$ satisfies

$$\|x_N - x_*\|_2^2 \leq c^N \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution, $0 < c < 1$ and $c = e^{-\frac{1}{e}\frac{\mu}{L}}$.

Theorem 2.18 ([Gli19b]). Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of GM with a fixed step size $\frac{1}{L}$ satisfies

$$\|\nabla f(x_N)\|_2^2 \leq d^N \|\nabla f(x_0)\|_2^2,$$

where $0 < d < 1$ and $d = e^{-\frac{1}{e}\frac{\mu^2}{L^2}}$

Remark 2.19. In this master thesis, the ratio $\frac{1}{e}\frac{\mu}{L}$ is always much smaller than one and therefore, one can approximate the constant c in Theorems [2.16], [2.17] and d in Theorem [2.18] respectively as $c = (1 - \frac{1}{e}\frac{\mu}{L})$ and $d = (1 - \frac{1}{e}\frac{\mu^2}{L^2})$ since $e^x \approx 1 + x$ for $x \ll 1$.

Theorems [2.16], [2.17] and [2.18] guarantee that the gradient descent for smooth strongly convex functions converges in $\mathcal{O}(\alpha^N)$ with $0 < \alpha < 1$ for the three performance criteria considered. This means that to achieve a performance guarantee of $f(x_N) - f(x_*) \leq \epsilon$ or $\|x_N - x_*\|_2^2 \leq \epsilon$, $\mathcal{O}(\frac{L}{\mu} \log \frac{1}{\epsilon})$ iterations of the gradient descent must be performed since $\alpha = c$ in that case and to achieve a performance guarantee of $\|\nabla f(x_N)\|_2^2 \leq \epsilon$, $\mathcal{O}(\frac{L^2}{\mu^2} \log \frac{1}{\epsilon})$ iterations of the gradient descent must be performed since $\alpha = d$ in that case. For these performance criteria, the convergence rate is **linear** since $\nu \in (0, 1)$. Therefore, the convergence rate of the gradient method is better for smooth strongly convex functions than for smooth convex functions.

Finally, the performance bounds from Theorems [2.16], [2.17] and [2.18] are for the classical GM on L -smooth μ -strongly convex functions, which are better conditioned functions than the algorithm was designed for, i.e., L -smooth convex functions. A more adapted version of the gradient method for L -smooth μ -strongly convex functions is to take the step size $h = \frac{2}{\mu+L}$ but this version requires the knowledge of μ which is complicated to estimate in practice. [Nes18, Section 1.2.3]

2.3.2 Fast gradient method - FGM

There exist many versions of the fast gradient method, also called the fast gradient descent. We have chosen to implement the fast gradient method introduced by Nesterov in the mid 80s [Nes83]: this method accelerates the gradient method by using a momentum parameter θ . The algorithm is very similar to a standard gradient method, the only difference is that the gradient is taken at a point x_i that is an extrapolation of the iterate y_i along the direction $y_i - y_{i-1}$: the point x_i is then a θ -combination of the points y_i and y_{i-1} . The iteration rule of the fast gradient method is represented in Figure [2.3].

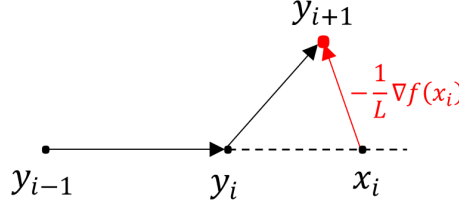


Figure 2.3: x_i is an extrapolation of y_i along the direction $y_i - y_{i-1}$. The gradient of f is evaluated at x_i and the new iterate y_{i+1} , i.e., the red point, is defined as $x_i - \frac{1}{L}\nabla f(x_i)$.

Unlike the gradient method, the iterates of the fast gradient method are not monotone and thus, the function values can increase during several iterations before decreasing again. The algorithm is implemented as follows.

FGM

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$ and $\theta_0 = 1$

For $i = 0, \dots, N-1$

$$\begin{aligned} y_{i+1} &= x_i - \frac{1}{L}\nabla f(x_i) \\ \theta_{i+1} &= \frac{1 + \sqrt{4\theta_i^2 + 1}}{2} \\ x_{i+1} &= y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}}(y_{i+1} - y_i) \end{aligned}$$

Convergence rates for smooth convex functions

Theorem 2.20 ([KF15, theorem 1]). *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth convex function. Then the iterate x_N of FGM satisfies*

$$f(x_N) - f(x_*) \leq \frac{2L}{(N+2)^2} \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution.

From previous Theorem 2.20 and using the Descent Lemma 2.12, the following theorem can be established.

Theorem 2.21. *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth convex function. Then the iterate x_N of FGM satisfies*

$$\|\nabla f(x_N)\|_2^2 \leq \frac{4L^2}{(N+2)^2} \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution.

Theorems 2.20 and 2.21 guarantee that the fast gradient descent for smooth convex functions converges in $\mathcal{O}(\frac{1}{N^2})$ for the two performance criteria $f(x_N) - f(x_*)$ and $\|\nabla f(x_N)\|_2^2$.

This means that to achieve a performance guarantee of $f(x_N) - f(x_*) \leq \epsilon$ or $\|\nabla f(x_N)\|_2^2 \leq \epsilon$, $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ iterations of the fast gradient descent must be performed. For these performance criteria, the convergence rate is **sublinear** since $\nu = 1$.

Then, Remark 2.15 also applies to the convergence rate of smooth convex functions for the fast gradient descent. Therefore, only a weak guarantee can be provided for the performance criterion $\|x_N - x_*\|_2^2$ for smooth convex functions. The convergence rate for this performance criterion is also **sublinear** since $\nu = 1$ but there is no guarantee on the number of iterations of the fast gradient method to be performed in order to reach a performance guarantee of $\|x_N - x_*\|_2^2 \leq \epsilon$.

Convergence rates for smooth strongly convex functions

By combining Remark 2.6 for strongly convex functions and Theorem 2.20, the two following theorems can be obtained.

Theorem 2.22. *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of FGM satisfies*

$$f(x_N) - f(x_*) \leq \frac{4L}{\mu(N+2)^2} (f(x_0) - f(x_*)),$$

where x_* is any optimal solution.

Theorem 2.23. *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of FGM satisfies*

$$\|x_N - x_*\|_2^2 \leq \frac{4L}{\mu(N+2)^2} \|x_0 - x_*\|_2^2,$$

where x_* is any optimal solution.

By combining Remark 2.6 for strongly convex functions and Theorem 2.21, the following theorem can be obtained:

Theorem 2.24. *Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a L -smooth μ -strongly convex function. Then the iterate x_N of FGM satisfies*

$$\|\nabla f(x_N)\|_2^2 \leq \frac{4L^2}{\mu^2(N+2)^2} \|\nabla f(x_0)\|_2^2.$$

Theorems 2.22, 2.23 and 2.24 guarantee that the fast gradient descent for smooth strongly convex functions converges in $\mathcal{O}(\frac{1}{N^2})$ for the three performance criteria considered. This means that to achieve a performance guarantee of $f(x_N) - f(x_*) \leq \epsilon$ or $\|x_N - x_*\|_2^2 \leq \epsilon$ or $\|\nabla f(x_N)\|_2^2 \leq \epsilon$, $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ iterations of the fast gradient descent must be performed. For these performance criteria, the convergence rate is **sublinear** since $\nu = 1$. Therefore, the convergence rate of the fast gradient method is the same for smooth strongly convex functions as for smooth convex functions.

Moreover, we can observe that the performance bounds for smooth strongly convex functions are better for the gradient method (Theorems 2.16, 2.17 and 2.18) than for the fast

gradient method (Theorems 2.22, 2.23 and 2.24). Then, the convergence rate for the gradient method is linear while it is sublinear for the fast gradient method. However, since the performance bounds are upper bounds, a method can have a poor performance bound but can work very well in practice. Thus, the performance bounds or the convergence rates do not allow to determine which method performs better.

Finally, the performance bounds from Theorems 2.22, 2.23 and 2.24 are for the classical FGM on L -smooth μ -strongly convex functions, which are better conditioned functions than the algorithm was designed for, i.e., L -smooth convex functions. A more adapted fast gradient method for L -smooth μ -strongly convex functions with better convergence rate will be examined in Chapter 5 but this method requires the knowledge of μ which is complicated to estimate in practice.

2.4 Conclusion

To conclude this chapter, we have seen that by assuming that the function f is L -smooth μ -strongly convex, i.e., $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, we can obtain guarantee on the worst-case performance of the gradient method and the fast gradient method for the three performance criteria considered in this master thesis. One notices that the performance bounds and the convergence rates are really helpful to evaluate the performance of optimization methods. However, a poor rate of convergence does not mean that the methods do not work properly or do not converge fast. Indeed, performance bounds are upper bounds and so the methods could converge much faster in practice than their theoretical upper bounds. Therefore, to better evaluate the worst-case performance of methods, we can use the toolbox PESTO that computes the exact worst-case performance of first-order optimization methods.

3

Performance estimation problems

This chapter presents the performance estimations problems based on the thesis of Adrien Taylor [Tay17] and his articles [THG16] and [THG17b] written with Julien Hendrickx and François Glineur. The performance estimation approach allows computing automatically the exact worst-case of fixed-step first-order methods for smooth convex (possibly strongly convex) unconstrained optimization. This approach can also be used to compare first-order optimization methods and to determine the best one in a more precise way than with performance bounds. Indeed, as mentioned in Chapter 2, if a method has a bad performance bound, it does not mean that this method does not work well or converges slowly.

Throughout this chapter, we will use a simple example to illustrate and motivate the theoretical concepts of performance estimation problems: obtaining the worst-case guarantees of a single gradient step on a smooth strongly convex function. This example is based on the article [THG17a].

3.1 Introduction to performance estimation

Let us remind that we want to solve the following unconstrained minimization problem

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

with f a smooth (strongly) convex function using iterative first-order black-box methods. Then a central question is «*how well does a first-order method perform?* ». As explained in Chapter 2, to answer this question, an approach is to study what happens in the worst-case: the goal is to determine the worst-case performance of solutions obtained with a specific first-order method for a given class of problems, after a given number of iterations. Then, the exact numerical value of the worst-case performance of a specific method is generally unknown since most of the papers focus on bounds on the worst-case performance which are usually not assessed precisely.

The performance estimation problem is a different approach that is focused on the computation of exact worst-case performance of a given first-order black-box method, on a given class of functions, after a given number of iterations.

To compute the exact worst-case performance, the problem of finding the worst-case performance of a black-box first-order method is expressed as an optimization problem over a set of smooth (strongly) convex functions and initial conditions. This approach was pioneered by Drori and Teboulle [DT12] who named it «performance estimation problem» and it was then developed by Adrien Taylor [Tay17]. Several papers were inspired by this approach, e.g., in [KGT17; Faz+18; KF15; LRP16; WRJ18], and it is therefore a rapidly developing field.

Formal definition

In order to formulate the problem of finding the exact worst-case performance as an optimization problem, we consider the following notations:

- $\mathcal{O}_f$ is an oracle that returns **first-order information** about specific points: in our case, we have $\mathcal{O}_f(x) = \{f(x), \nabla f(x)\}$.
- $\mathcal{M}$ is a first-order black-box **method** which only uses first-order information about the objective function with the oracle $\mathcal{O}_f$.
 $\mathcal{M}_i$ outputs the iterate after the i^{th} iteration of $\mathcal{M}$.
- N is the **number of iterations**.
- $x_* \in \mathbb{R}^d$ is an **optimal solution**, i.e., a minimizer of f .
- $\{x_0, x_1, \dots, x_N\} \in \mathbb{R}^{d \times (N+1)}$ are the **iterates** with x_0 the initial point. They are given by

$$\begin{aligned} x_1 &= \mathcal{M}_1(x_0, \mathcal{O}_f(x_0)), \\ x_2 &= \mathcal{M}_2(x_0, \mathcal{O}_f(x_0), \mathcal{O}_f(x_1)), \\ &\vdots \\ x_N &= \mathcal{M}_N(x_0, \mathcal{O}_f(x_0), \dots, \mathcal{O}_f(x_{N-1})). \end{aligned}$$

- $\mathcal{P}$ is the **performance criterion** to be minimized: this criterion allows to measure the performance of a given method $\mathcal{M}$ on a specific function f with a specific starting point. It depends only on the function f , the sequence of iterates $\{x_0, x_1, \dots, x_N\}$ generated by the method $\mathcal{M}$ and the optimal solution x_* : therefore, we have $\mathcal{P}(\mathcal{O}_f, x_0, \dots, x_N, x_*)$.
As previously mentioned in Chapter 2, the objective function accuracy $f(x_N) - f(x_*)$, the norm of the gradient $\|\nabla f(x_N)\|_2^2$ or the distance to the optimal solution $\|x_N - x_*\|_2^2$ are examples of performance criteria.
- $\mathcal{F}$ is a given **class of smooth (strongly) convex functions** over which we want to estimate the worst-case performance of the method $\mathcal{M}$.
- R is a normalization parameter used in the **initial condition**. Note that it is necessary to add an initial condition because, in most situations, the performance of a first-order method cannot be sensibly assessed without such a constraint. Indeed, without an initial condition, the worst-case could be unbounded. A bound on the objective function accuracy at the initial point $f(x_0) - f(x_*) \leq R$ or on the distance

between the initial point and the optimal solution $\|x_0 - x_*\|_2 \leq R$ or on the gradient norm at the initial point $\|\nabla f(x_0)\|_2 \leq R$ are examples of initial conditions. In this master thesis, we mostly consider the second initial condition.

Then, knowing all these notations, we can write the optimization problem that we have to solve in order to find the exact worst-case performance. Since the methods considered try to minimize the performance criterion $\mathcal{P}$, their worst-case performance is obtained by maximizing this performance criterion $\mathcal{P}$ over functions in $\mathcal{F}$:

$$\begin{aligned} & \sup_{f, x_0, \dots, x_N, x_*} \mathcal{P}(\mathcal{O}_f, x_0, \dots, x_N, x_*) & (\text{PEP}) \\ \text{such that } & f \in \mathcal{F}, \\ & x_* \text{ is optimal for } f, \\ & x_1, \dots, x_N \text{ is generated from } x_0 \text{ by the method } \mathcal{M} \text{ with } \mathcal{O}_f, \\ & \|x_0 - x_*\|_2 \leq R. \end{aligned}$$

Finite dimensional reformulation using interpolation

In the problem (PEP), we can see that there is an unknown function f as a variable and the problem (PEP) is therefore infinite-dimensional. In order to deal with this issue, one can observe that only the output of the oracle $\mathcal{O}_f$ at the iterates $\{x_0, x_1, \dots, x_N\}$ and at x_* is needed.

By defining a set $I = \{0, 1, \dots, N, *\}$ corresponding to the indices of the iterates and $\mathcal{O}_f(x_i) = \{f_i, g_i\}$ corresponding to the output of the oracle at each iterate x_i , we can reformulate the problem (PEP) by considering only the iterates $\{x_i\}_{i \in I}$, their function values $\{f_i\}_{i \in I}$ and their gradients $\{g_i\}_{i \in I}$:

$$\begin{aligned} & \sup_{\{x_i, f_i, g_i\}_{i \in I}} \mathcal{P}(\{x_i, f_i, g_i\}_{i \in I}) & (\text{f-PEP}) \\ \text{such that } & \exists f \in \mathcal{F} \text{ with } \mathcal{O}_f = \{f_i, g_i\} \forall i \in I, \\ & g_* = 0, \\ & x_1, \dots, x_N \text{ is generated from } x_0 \text{ by the method } \mathcal{M} \text{ with } \{f_i, g_i\}_{i \in \{0, \dots, N-1\}}, \\ & \|x_0 - x_*\|_2 \leq R. \end{aligned}$$

where optimality of x_* and constraint $g_* = 0$ are equivalent since the (Optimization Problem) is unconstrained.

This problem is therefore finite but there is still an issue: we need to cope with the first constraint. This constraint can be understood as requiring that the points $\{x_i, f_i, g_i\}_{i \in I}$ must be interpolable by a function belonging to the class $\mathcal{F}$. Thus, this constraint ensures that the optimal values of (PEP) and (f-PEP) are equal. Indeed, any solution of the problem (PEP) can be converted into a feasible solution for (f-PEP) and inversely. This issue will be solved in the next section.

Toy example: worst-case analysis of a gradient step - Part 1

Let us apply the theory described in this chapter to a simple example: a single gradient step. We want to answer the following question:

Let f be a L -smooth μ -strongly convex function, i.e., let $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$; what can we exactly guarantee on $\|\nabla f(x_1)\|_2^2$ when x_1 was obtained from a gradient step $x_1 = x_0 - \frac{1}{L}\nabla f(x_0)$ and when $\|\nabla f(x_0)\|_2^2 \leq R^2$?

In this case, the problem (PEP) can be written as

$$\begin{aligned} & \sup_{f, x_0, x_1} \|\nabla f(x_1)\|_2^2 && \text{(gradient-PEP)} \\ \text{such that } & f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d) && \text{Functional class,} \\ & x_1 = x_0 - \frac{1}{L}\nabla f(x_0) && \text{Method,} \\ & \|\nabla f(x_0)\|_2^2 \leq R^2 && \text{Initial condition.} \end{aligned}$$

An interesting observation is that this example does not include the solution x_* which is not computable in practice. Then, from this example (gradient-PEP), it is clear that the initial condition is crucial in order to avoid an unbounded solution. However, as explained above, this problem is infinite-dimensional. Therefore we need to reformulate it by considering only the points $\{x_i, f_i, g_i\}_{i \in \{0,1\}}$ in order to make it finite, which corresponds to the problem (f-PEP)

$$\begin{aligned} & \sup_{\{x_i, f_i, g_i\}_{i \in \{0,1\}}} \|g_1\|_2^2 && \text{(gradient-f-PEP)} \\ \text{such that } & \exists f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d) : f_i = f(x_i) \text{ and } g_i = \nabla f(x_i) \text{ for } i \in \{0,1\}, \\ & x_1 = x_0 - \frac{1}{L}g_0, \\ & \|g_0\|_2^2 \leq R^2. \end{aligned}$$

In the problem (gradient-f-PEP), it is easier to notice that the first constraint indeed requires that the function f in the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ can interpolate the points $\{x_0, f_0, g_0, x_1, f_1, g_1\}$.

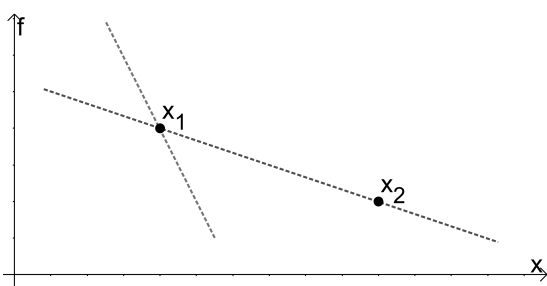
3.2 Smooth strongly convex interpolation

In this section, we will give a necessary and sufficient condition that ensures that the set of values $\{x_i, f_i, g_i\}_{i \in I}$ can be interpolated by a function belonging to a class $\mathcal{F}$. More precisely, we focus on proper closed convex functions (i.e., whose epigraphs¹ are non-empty closed convex sets) belonging to $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ with $0 \leq \mu < L < +\infty$. Then, this condition will allow us to transform the problem (f-PEP) into a tractable one.

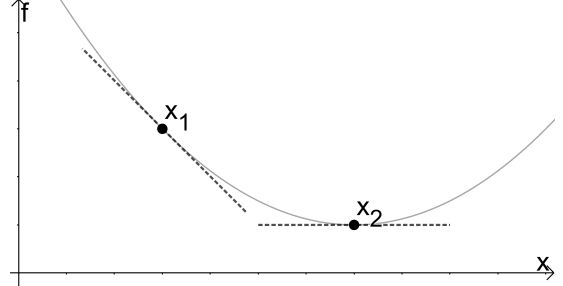
First, let us remind what interpolation is with the following definition and graphical examples on Figure 3.1.

¹The *epigraph* of a function $f : X \rightarrow \mathbb{R}$ is the set $\{(x, t) \in X \times \mathbb{R} : f(x) \leq t \text{ and } x \in \text{dom } f\}$.

Definition 3.1 ($\mathcal{F}_{\mu,L}$ -interpolation). Let I be an index set and consider the set of triples $S = \{(x_i, f_i, g_i)\}_{i \in I}$ where $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. The set S is $\mathcal{F}_{\mu,L}$ -interpolable if there exists a function $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that we have both $f_i = f(x_i)$ and $g_i = \nabla f(x_i)$ for all $i \in I$.



(a) **Impossible interpolation:** the set of triples $\{x_1, f_1, g_1\} = \{2, 2, -2\}$ and $\{x_2, f_2, g_2\} = \{5, 1, -\frac{1}{3}\}$ cannot satisfy both the smoothness and the convexity requirements.



(b) **Possible interpolation:** the set of triples $\{x_1, f_1, g_1\} = \{3, 3, -1\}$ and $\{x_2, f_2, g_2\} = \{7, 1, 0\}$ can be interpolated by a smooth convex function, actually by a quadratic function.

Figure 3.1: Examples of a set of triples that cannot or can be interpolated by a L -smooth μ -strongly convex function.

Then, the necessary and sufficient condition that ensures a smooth strongly convex interpolation is given by the following theorem.

Theorem 3.2 ($\mathcal{F}_{\mu,L}$ -interpolability). *Set $\{x_i, f_i, g_i\}_{i \in I}$ is $\mathcal{F}_{\mu,L}$ -interpolable if and only if the following set of conditions holds for every pair of indices $i \in I$ and $j \in I$*

$$f_i - f_j - g_j^\top (x_i - x_j) \geq \frac{1}{2(1 - \mu/L)} \left(\frac{1}{L} \|g_i - g_j\|_2^2 + \mu \|x_i - x_j\|_2^2 - 2 \frac{\mu}{L} (g_j - g_i)^\top (x_j - x_i) \right). \quad (3.1)$$

More details about how this condition was achieved and proofs are available in [\[Tay17\]](#).

From this theorem, one can reformulate the problem [\(f-PEP\)](#). In order to simplify the derivations, we can assume that $x_* = 0$ and $f_* = 0$ without loss of generality since we consider the class of function $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ and first-order methods that are invariant to additive shifts in the function values and translations in their domain. Therefore, the following problem [\(d-PEP\)](#) can be defined.

$$\sup_{\{x_i, f_i, g_i\}_{i \in I} \in (\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R})^{N+2}} \mathcal{P}(\{x_i, f_i, g_i\}_{i \in I}) \quad (\text{d-PEP})$$

such that $\{x_i, f_i, g_i\}_{i \in I}$ is $\mathcal{F}_{\mu,L}$ -interpolable,

$$\{x_*, f_*, g_*\} = \{0, 0, 0\},$$

$x_1, \dots, x_N$ is generated from x_0 by the method $\mathcal{M}$ with $\{f_i, g_i\}_{i \in \{0, \dots, N-1\}}$,

$$\|x_0 - x_*\|_2 \leq R.$$

The problem (d-PEP) is thus a particular case of (f-PEP) where the function class $\mathcal{F}$ is set to $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$.

However, there is still an issue with this formulation: the problem (d-PEP) is not convex since it contains some non-convex quadratic constraints (e.g., $g_j^\top x_i$ is not convex in the interpolation condition (3.1)). As the class of non-convex problems is in general NP-hard, some tricks will be needed to resolve this issue: the next section will explain how to transform this problem into a convex one for a certain class of first-order black-box methods.

Toy example: worst-case analysis of a gradient step - Part 2

Let us come back to our example: we want to compute the worst-case performance of a single gradient step for a L -smooth μ -strongly convex function. Using the smooth strongly convex interpolation Theorem (3.2), the problem (gradient-f-PEP) becomes:

$$\begin{aligned} & \sup_{\{x_i, f_i, g_i\}_{i \in \{0,1\}}} \|g_1\|_2^2 && \text{(gradient-d-PEP)} \\ & \text{such that interpolation constraints (3.1) hold for all } i, j \in \{0, 1\}, \\ & x_1 = x_0 - \frac{1}{L}g_0, \\ & \|g_0\|_2^2 \leq R^2. \end{aligned}$$

3.3 Convex formulation for performance estimation

In order to obtain a convex formulation of the problem (d-PEP), we restrict ourselves to the class of fixed-step first-order methods.

Definition 3.3 (fixed-step method). A method $\mathcal{M}$ is a *fixed-step* method if its iterates are computed according to

$$x_i = x_0 - \sum_{k=0}^{i-1} h_{i,k} g_k,$$

where $h_{i,k}$ is a scalar coefficient.

A fixed-step method applying N steps is thus entirely defined by the lower triangular matrix $H = \{h_{i,k}\}_{1 \leq i \leq N, 0 \leq k \leq N-1} \in \mathbb{R}^{N \times N}$ (where $h_{i,k}$ is equal to zero if $k \geq i$). The gradient method and the fast gradient method are included in this class of methods.

Then, let us define some notations:

- $P = [g_0 \ g_1 \ \cdots \ g_N \ x_0] \in \mathbb{R}^{d \times N+2}$.

- $G = P^\top P \in \mathbb{S}^{N+2}$ is the Gram matrix^[2], which is equivalent to

$$G = \{G_{i,j}\}_{0 \leq i,j \leq N+1} \text{ with } \begin{cases} G_{i,j} = g_i^\top g_j & \text{for any } 0 \leq i, j \leq N, \\ G_{N+1,j} = x_0^\top g_j & \text{for any } 0 \leq j \leq N, \\ G_{i,N+1} = g_i^\top x_0 & \text{for any } 0 \leq i \leq N, \\ G_{N+1,N+1} = x_0^\top x_0. \end{cases}$$

This Gram matrix has the following characteristics.

- The constraints in the problem (d-PEP) can be entirely formulated in terms of the entries of the Gram matrix G and the function values f_i since all iterates can be obtained from x_0 using Definition 3.3.
- $G \succeq 0$, i.e., the Gram matrix G is a positive semidefinite matrix.
- $\text{rank } G \leq d$ since its elements are computed from vectors x_0 and g_i that all belong to $\mathbb{R}^d$.
- $h_i \in \mathbb{R}^{N+2}$ for any $i \in \{0, \dots, N-1\}$ and $h_* \in \mathbb{R}^{N+2}$

$$h_i^\top = [-h_{i,0} \ -h_{i,1} \ \dots \ -h_{i,i-1} \ 0 \ \dots \ 1] \text{ and } h_*^\top = [0 \ \dots \ 0],$$
such that $x_i = Ph_i$ (see Definition 3.3).
- $u_i = e_{i+1} \in \mathbb{R}^{N+2}$ are the canonical basis vectors and $u_* \in \mathbb{R}^{N+2}$ is the vector of zeros.

These notations allow us to rewrite the interpolation constraints (3.1) from Theorem (3.2) as follows for all $i, j \in I$:

$$\begin{aligned} f_i \geq f_j &+ \frac{L}{L-\mu}(u_j^\top Gh_i - u_j^\top Gh_j) + \frac{1}{2(L-\mu)}(u_i - u_j)^\top G(u_i - u_j) \\ &+ \frac{\mu}{L-\mu}(u_i^\top Gh_j - u_i^\top Gh_i) + \frac{L\mu}{2(L-\mu)}(h_i - h_j)^\top G(h_i - h_j). \end{aligned}$$

These constraints as well as the initial condition $\|x_0 - x_*\|_2 \leq R$ can also be formulated using the trace operator. To this end, we define the matrices A_{ij} for all $i, j \in I$ and A_R as follows:

$$\begin{aligned} 2A_{ij} &= \frac{L}{L-\mu} \left(u_j(h_i - h_j)^\top + (h_i - h_j)u_j^\top \right) + \frac{1}{L-\mu} (u_i - u_j)(u_i - u_j)^\top \\ &+ \frac{\mu}{L-\mu} \left(u_i(h_j - h_i)^\top + (h_j - h_i)u_i^\top \right) + \frac{L\mu}{L-\mu} (h_i - h_j)(h_i - h_j)^\top, \\ A_R &= u_{N+1}u_{N+1}^\top. \end{aligned} \tag{3.2}$$

Then, we can rewrite all the constraints of the problem, i.e., the interpolation constraints, the initial condition and the constraints on G , into a compact form:

$$\begin{aligned} f_j - f_i + \text{Tr}(GA_{ij}) &\leq 0 \quad \text{for all } i, j \in I, \\ \text{Tr}(GA_R) - R^2 &\leq 0, \\ G &\succeq 0, \\ \text{rank } G &\leq d. \end{aligned}$$

² $\mathbb{S}^n$ is the set of symmetric matrices of size $n \times n$.

Note that all these constraints except the rank constraint are linear in their variables $f \in \mathbb{R}^{N+1}$ and $G \in \mathbb{S}^{N+2}$. Therefore, if we consider a performance criterion $\mathcal{P}$ that is a concave semidefinite-representable function in G and f , the worst-case estimation problem can be cast into a convex semidefinite optimization problem plus a rank constraint. In particular, classical performance criteria such as $f(x_N) - f(x_*)$, $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$ are linear in f and G and thus are suitable.

In this section, we will focus on linear performance criterion:

$$\mathcal{P} = b^\top f + \text{Tr}(CG)$$

where $b \in \mathbb{R}^{N+1}$ and $C \in \mathbb{S}^{N+2}$. But other concave performance criteria could be used. More precisely, the following proposition states the condition for which a performance estimation problem can be formulated as a semidefinite program (sdp).

Proposition 3.4 ([THG17b], proposition 2.6). *Let $\mathcal{F}$ be a class of functions for which the interpolations constraints (3.1) can be formulated using a finite number of convex constraints involving only G and f . Let $\mathcal{M}$ be a first-order method, $\mathcal{P}$ be a performance criterion and $\mathcal{I}$ be an initial condition such that these objects can be expressed finitely in terms of G and f . In brief, all objects $\mathcal{F}$, $\mathcal{M}$, $\mathcal{P}$ and $\mathcal{I}$ are said to be Gram-representable. Then, computing the worst-case performance $\mathcal{P}$ of a method $\mathcal{M}$ after N iterations on a function belonging to class $\mathcal{F}$ with an initial condition $\mathcal{I}$ can be formulated as a sdp with G and f as variables.*

Finally, let us consider the class $\mathcal{F}_{\mu,L}(\mathbb{R}^d)$ of L -smooth μ -strongly convex functions with $0 \leq \mu < L < +\infty$, a fixed-step first-order method $\mathcal{M}$ that performs N steps described by the matrix $H \in \mathbb{R}^{N \times N}$ and a performance criterion $\mathcal{P}$ that depends linearly on the function values at the iterates and quadratically on these iterates and their gradients. Then, the worst-case performance after N iterations of the method $\mathcal{M}$ is obtained by resolving the following rank-constrained semidefinite program:

$$\begin{aligned} & \sup_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} b^\top f + \text{Tr}(CG) && (\text{sdp-PEP}(d)) \\ & \text{such that } f_j - f_i + \text{Tr}(GA_{ij}) \leq 0, && i, j \in I \\ & \text{Tr}(GA_R) - R^2 \leq 0, \\ & G \succeq 0, \\ & \text{rank } G \leq d. \end{aligned}$$

Note that since G has a size $N + 2$, it has a rank at most equal to $N + 2$. Therefore, if $N + 2 \leq d$, the optimal value of (sdp-PEP(d)) is equal to the optimal value of the following convex semidefinite program:

$$\begin{aligned}
& \sup_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} b^\top f + \text{Tr}(CG) & (\text{sdp-PEP}) \\
& \text{such that } f_j - f_i + \text{Tr}(GA_{ij}) \leq 0, \quad i, j \in I \\
& \quad \text{Tr}(GA_R) - R^2 \leq 0, \\
& \quad G \succeq 0.
\end{aligned}$$

Solving this convex relaxation allows obtaining worst-case guarantees independently of the value of d . Moreover, we have the guarantee that the optimal value of (sdp-PEP) can be achieved by a L -smooth μ -strongly convex function of dimension at most $N + 2$. Therefore, if by solving (sdp-PEP), we obtain a function of dimension $d' \leq N + 2$, the resulting optimal value corresponds to the exact worst-case performance for all functions of dimensions greater or equal than d' and is an upper bound for the functions of dimensions smaller than d' . Hence, the best possible guarantee with no dependence on d is given by the optimal value of (sdp-PEP) which is valid for any value of d .

Thanks to this convex formulation, we are now finally able to solve numerically the problem (sdp-PEP) and achieve global optimality.

Toy example: worst-case analysis of a gradient step - Part 3

Finally, let us find the worst-case performance of our example: a single gradient step. The gradient step is indeed a fixed-step method where $h_{1,0} = \frac{1}{L}$ in our example (see Definition 3.3).

- The variable x_1 can be expressed using the variables x_0 and g_0 : $x_1 = x_0 - \frac{1}{L}g_0$.
- The gram matrix is given by

$$G = \begin{pmatrix} \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle & \langle g_0, x_0 \rangle \\ \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle & \langle g_1, x_0 \rangle \\ \langle x_0, g_0 \rangle & \langle x_0, g_1 \rangle & \langle x_0, x_0 \rangle \end{pmatrix} \succeq 0.$$

The problem (gradient-d-PEP) can then be reformulated as an equivalent rank-constrained semidefinite program:

$$\begin{aligned}
& \sup_{G \in \mathbb{S}^3, f \in \mathbb{R}^2} G_{1,1} & (\text{gradient-sdp-PEP}(d)) \\
& \text{such that } f_j - f_i + \text{Tr}(GA_{ij}) \leq 0, \quad i, j \in \{0, 1\}, \\
& \quad G_{0,0} - R^2 \leq 0, \\
& \quad G \succeq 0, \\
& \quad \text{rank } G \leq d.
\end{aligned}$$

where the matrices A_{ij} and A_R are given above by the formulas (3.2). In order to solve numerically the problem (gradient-sdp-PEP(d)), we solve the convex relaxation which is independent of the value of d :

$$\begin{aligned}
& \sup_{G \in \mathbb{S}^3, f \in \mathbb{R}^2} G_{1,1} && \text{(gradient-sdp-PEP)} \\
& \text{such that } f_j - f_i + \text{Tr}(GA_{ij}) \leq 0, && i, j \in \{0, 1\}, \\
& G_{0,0} - R^2 \leq 0, \\
& G \succeq 0.
\end{aligned}$$

Note that (gradient-sdp-PEP) is equivalent to (gradient-sdp-PEP(d)) if $d \geq 3$. Moreover, we have the guarantee that the optimal value will be achieved by a function of dimension at most 3.

Finally, since the one step gradient is a quite simple method, it is possible to derive an analytical optimal value which is given by

$$\max\{\|\nabla f(x_1)\|_2^2\} = (1 - \frac{\mu}{L})^2 R^2 \quad [THG20a]. \quad (3.3)$$

As can be seen on Table 3.1, this theoretical result corresponds to the optimal value obtained from (gradient-sdp-PEP).

Table 3.1: Comparison between the theoretical value from (3.3) and the numerical value obtained by solving (gradient-sdp-PEP) with $L = 1$ and $R = 1$.

μ	Theoretical value	Numerical value
0.1	0.81	0.8100
0.01	0.9801	0.9801
0.001	0.9980	0.9980

The relaxation of the rank constraint had therefore no impact in this case. In addition, the corresponding worst-case functions are $f(x) = \frac{\mu}{2}|x|^2$ and $f(x) = \frac{L}{2}|x|^2$. Since these functions are 1-dimensional, the worst-case performance is exact for any value of d .

3.4 PESTO toolbox

The performance estimation problem explained in the previous sections is implemented in PESTO, a MATLAB toolbox. This toolbox provides a framework in which the methods can be described quite easily and then, PESTO takes care of the worst-case analysis and computation based on the performance estimation approach. More explanation, details and examples about the working procedure of PESTO are available in [THG20b] and more concisely in [THG17a].

Let us quote the advantages and disadvantages of this toolbox.

Benefits

On the one hand, this toolbox is quite easy to use and very intuitive: the user can describe the algorithms nearly as he would have implemented it instead of performing

potentially complicated and demanding semidefinite program modelling steps required by the performance estimation approach. It is therefore quite easy to obtain exact worst-case performance for a wide class of fixed-step first-order optimization methods and these worst-case guarantees can thus not be improved. This allows to ease the analysis of these methods: for example, proofs that could barely be obtained by hand can be reached thanks to PESTO.

Drawbacks

On the other hand, this toolbox has some disadvantages.

First, although PESTO can calculate the worst-case performance of a large class of first-order methods, not all methods can be implemented with PESTO. More precisely, methods whose step size depends on the variables, i.e., on the iterates, the function and the gradient values at those iterates, cannot be implemented with PESTO since these methods do not have a fixed step (see Definition (3.3)). Thus, PESTO cannot deal with If conditions on variables that influence the step size.

Moreover, PESTO cannot deal either with strict constraints. Indeed, the set on which we minimize our optimization problem must be closed and this is not possible with a strict condition.

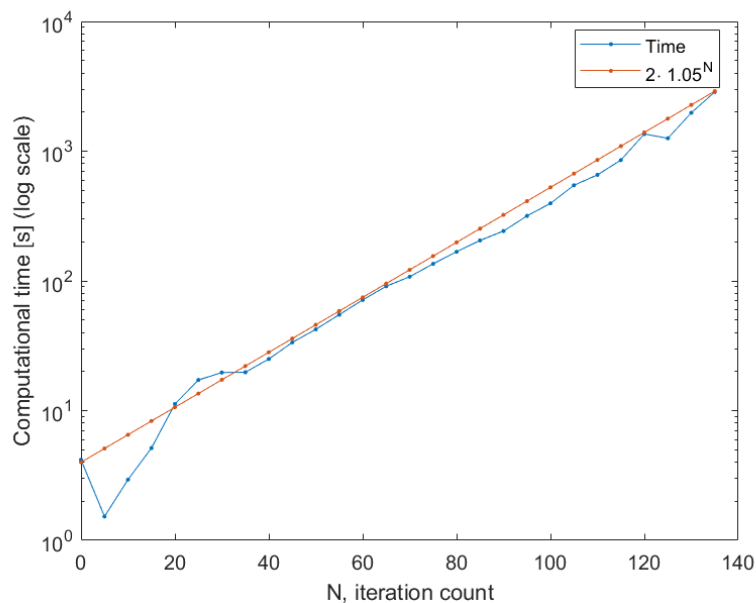


Figure 3.2: Computational time needed by PESTO to solve the performance estimation problem using FGM for N iterations with $\|x_N - x_*\|_2^2$ as performance criteria, $\|x_0 - x_*\|_2^2 \leq R^2$ as initial condition, $L = 1$, $R = 1$ and $\mu = 0.001$.

Then, when the number of iterations N increases, the problem to solve quickly becomes quite large. Indeed the number of constraints increases quadratically with the number of iterates N because of the constraint (3.1). Therefore, the computational time also increases quickly with N as can be seen in Figure 3.2. We can observe that the computa-

tional time evolution is close to a line on a semi-logarithmic plot. This means that there is an exponential relation between the time and N , i.e., $\text{time} = \alpha \cdot \beta^{\gamma N}$. With $\alpha = 2$, $\beta = 1.05$ and $\gamma = 1$, we obtain a line close to the computational time. Moreover, because of memory limitation, we cannot solve a problem with more than 150 iterations with our computers. It is possible to solve this problem for a higher number of iterations by using a computer or a server with more memory. However, even by using a server, we consider a maximum of 200 iterations due to the computational time limit.

Finally, even if we have the exact worst-case performance, proofs and analytical analysis might be hard to intuit or to generalize.

3.5 Conclusion

In this chapter, we have seen that the problem of finding the exact worst-case performance of a black-box first-order optimization method can be formulated as an optimization problem over a set of smooth (possibly strongly) convex functions and initial conditions for some specific performance criterion. Then, by using several tricks, we have shown that this problem can be expressed in a convex semidefinite program which can be solved numerically but only for fixed-step first-order methods. This approach is implemented in the toolbox PESTO that provides a framework in which methods can be described quite easily. However, this toolbox cannot deal with If conditions on variables that influence the step size or strict inequalities. Moreover, the time for solving the problem of finding the worst-case performance using PESTO increases exponentially with the number of iterations applied by the method considered.

4

Motivation

The main goal of this chapter is to study the performance and convergence of the gradient method (GM) and the fast gradient method (FGM) and to determine whether the property of strong convexity improves the performance and the convergence of these methods although they are not designed for strongly convex functions.

In Chapter 2, we have seen from a theoretical point of view that for smooth strongly convex functions, the performance bounds and the convergence rates are better for the gradient method than for the fast gradient method. However, the theoretical performance bounds do not allow to decide whether a method is better than another method since these bounds are not necessarily tight. Therefore, to determine whether the gradient method or the fast gradient method benefits from the strong convexity property, we compare the exact worst-case performance and convergence of these methods using the toolbox PESTO for smooth convex functions, i.e., $\mu = 0$, and smooth strongly convex functions, i.e., $\mu > 0$.

To evaluate and analyze the convergence of these methods, we calculate the ratio between the performance criterion at one iteration and the performance criterion at the previous iteration: more precisely, we compute $\frac{f(x_N) - f(x_*)}{f(x_{N-1}) - f(x_*)}$, $\frac{\|x_N - x_*\|_2^2}{\|x_{N-1} - x_*\|_2^2}$ and $\frac{\|\nabla f(x_N)\|_2^2}{\|\nabla f(x_{N-1})\|_2^2}$. Therefore, as N increases, these ratios tend respectively to the convergence rate ν for the performance criterion $f(x_N) - f(x_*)$ and to the square of the convergence rate ν^2 for the two performance criteria $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$ (see Definition 2.11).

In this master thesis, we consider the class of 1-smooth μ -strongly convex functions and the initial condition $\|x_0 - x_*\|_2^2 \leq 1$. Moreover, in this chapter, results are shown for the two performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. The same kind of results are observed for the performance criterion $\|\nabla f(x_N)\|_2^2$, as shown in Appendix 4.A.

4.1 Worst-case performance and convergence of GM

First, it can be seen on Figures 4.1b and 4.2b that the convergence rate of GM is linear for smooth strongly convex functions since the ratio $\frac{f(x_N) - f(x_*)}{f(x_{N-1}) - f(x_*)}$ or $\frac{\|x_N - x_*\|_2^2}{\|x_{N-1} - x_*\|_2^2}$ tends to a constant belonging to $(0, 1)$ (see Definition 2.11). This confirms the theoretical convergence rate we derived from Theorems 2.16 and 2.17. However, when the function is

convex, i.e., when $\mu = 0$, these ratios tend to or are equal to 1 for all performance criteria. It confirms the theoretical convergence rate we derived from Theorem 2.13 and Remark 2.15: the convergence rate of GM is sublinear for smooth convex functions. Thus, the strong convexity property significantly improves the convergence of the gradient method.

Moreover, Figures 4.1a and 4.2a show that the bigger the μ , the better the worst-case performance is. Similarly, we can see in Figures 4.1b and 4.2b that the biggest μ provides the lower rate of convergence, which means that functions with a high μ converge more quickly. Therefore, it shows that the gradient method takes advantage of the strong convexity parameter.

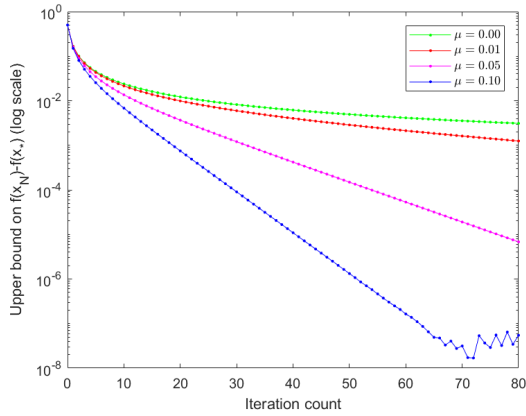
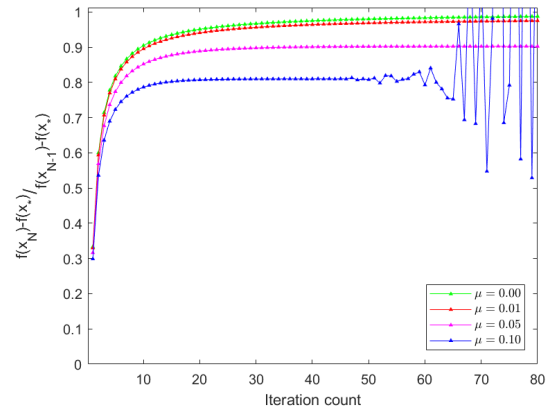
(a) Worst-case performance for several μ (b) Convergence for several μ

Figure 4.1: Comparison of the worst-case performance and the convergence of GM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $f(x_N) - f(x_*)$ as performance criterion.

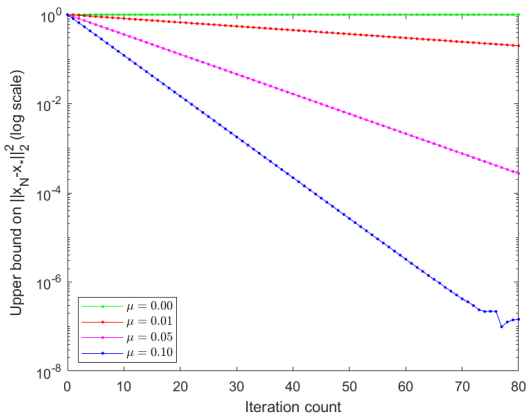
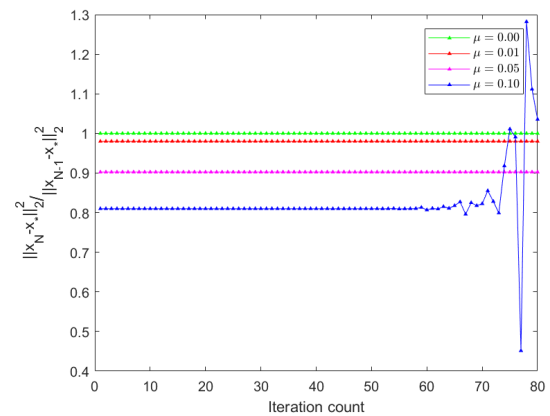
(a) Worst-case performance for several μ (b) Convergence for several μ

Figure 4.2: Comparison of the worst-case performance and the convergence of GM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $\|x_N - x_*\|_2^2$ as performance criterion.

Note that for larger μ , i.e., for $\mu = 0.1$, and after a certain number of iterations, i.e., after 60 iterations, we observe in Figures 4.1a and 4.2a that the worst-case performance stagnates by oscillating around a certain value due to numerical errors. Indeed, in these cases, the performance is close to zero, i.e., is less than 10^{-6} , and the accuracy of the solver is then reached which causes the appearance of numerical errors. Thus, the ratios explode after 60 iterations as shown in Figures 4.1b and 4.2b.

Afterwards, for the performance criterion $\|x_N - x_*\|_2^2$, the performance in Figure 4.2a and the ratios in Figure 4.2b are constant for $\mu = 0$. Indeed, this confirms Remark 2.15: the worst-case function can be a horizontal line and at each iteration, we have $x_{i+1} = x_i$ leading to a stagnation in the gradient descent. Thus, the worst-case performance is given by $\|x_N - x_*\|_2^2 = \|x_0 - x_*\|_2^2 = 1$ for smooth convex functions.

4.2 Worst-case performance and convergence of FGM

Figures 4.3a and 4.4a show that the worst-case performance of the fast gradient method starts by decreasing rapidly and then, more slowly until it stagnates after a certain number of iterations. For example, for $f(x_N) - f(x_*)$ and $\mu = 0.1$, the performance starts by decreasing rapidly until 20 iterations and then, more slowly until it stagnates after 50 iterations. It leads to a ratio $\frac{f(x_N) - f(x_*)}{f(x_{N-1}) - f(x_*)}$ or $\frac{\|x_N - x_*\|_2^2}{\|x_{N-1} - x_*\|_2^2}$ that varies and oscillates much more, especially for big μ , but seems to tend on average towards 1, as observed in Figures 4.3b and 4.4b. This means that the convergence rate of the fast gradient method is sublinear both for strongly convex and non strongly convex functions for all considered performance criteria. This confirms the theoretical convergence rates we derived from Theorems 2.20, 2.22, 2.23 and Remark 2.15 given in Chapter 2.

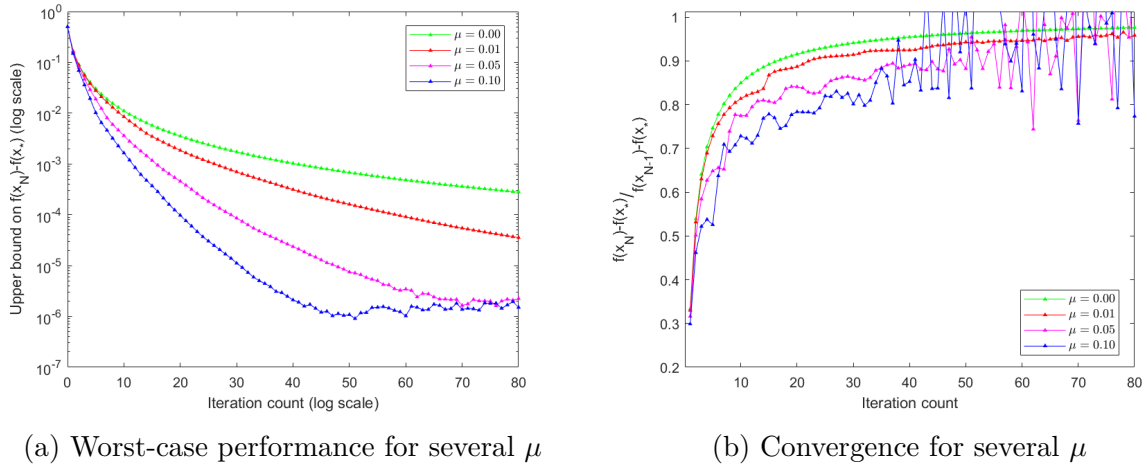


Figure 4.3: Comparison of the worst-case performance and the convergence of FGM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $f(x_N) - f(x_*)$ as performance criterion.

Therefore, the strong convexity property does not improve much the performance and

the convergence of the fast gradient method. This could come from the fact that, unlike the gradient descent, fast gradient methods are not guaranteed to be monotone in the objective function values since there is no guarantee such as the Descent Lemma 2.12 for the fast gradient method. That is the function values may increase for a couple of iterates, before decreasing again and thus, oscillations of the iterates appear around the solution. This could lead to slower convergence. [GB14]

Then, in Figure 4.4a, we can see that rebounds occur in the worst-case performance evolution which can also significantly slow down the convergence of the fast gradient method.

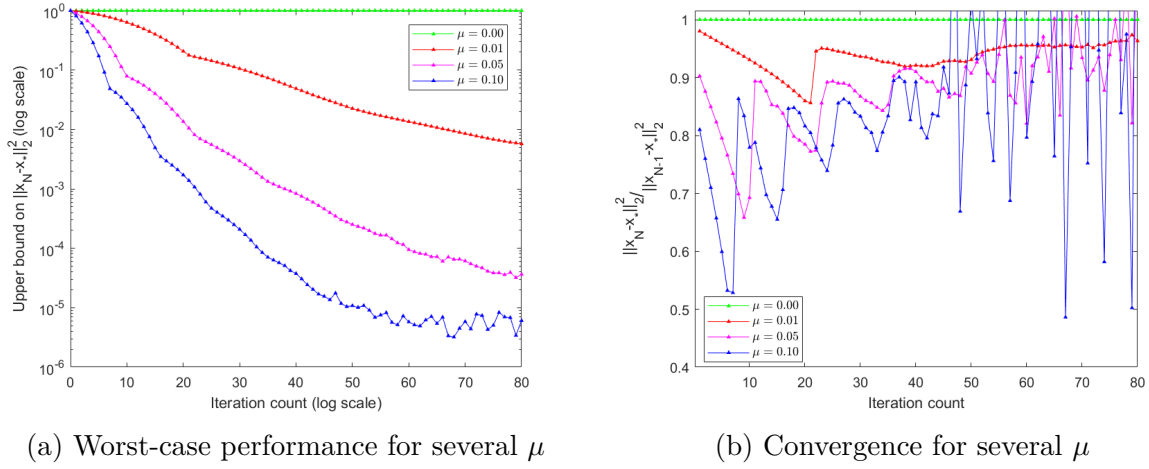


Figure 4.4: Comparison of the worst-case performance and the convergence of FGM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $\|x_N - x_*\|_2^2$ as performance criterion.

Moreover, as previously mentioned, Figures 4.3b and 4.4b show that the ratio starts to explode after a certain number of iterations: for example, for $f(x_N) - f(x_*)$ and $\mu = 0.1$, the ratio $\frac{f(x_N) - f(x_*)}{f(x_{N-1}) - f(x_*)}$ starts to explode after 40 iterations. This can be explained by the fact that the worst-case performance, as observed in Figures 4.3a and 4.4a, is close to zero and thus, numerical errors can occur.

4.3 Comparison between GM and FGM

In this section, let us compare the performance of both the gradient method and the fast gradient method for several values of the parameter μ .

In a general way, we observe in Figure 4.5a that for smooth convex functions, the fast gradient method is better than the gradient method. However, Figures 4.5a and 4.5b show that the gradient method for strongly convex functions is better than the fast gradient method after a certain number of iterations for both performance criteria. Indeed, for smooth strongly convex functions, the convergence rate of the gradient method is linear while the convergence rate of the fast gradient method is sublinear and as illustrated in

Figure 2.2 from Chapter 2, the linear convergence is slower at first but later becomes better than the sublinear convergence.

Moreover, we can see that the higher the μ , the quicker the gradient method becomes better than the fast gradient method. For example, we observe in Figure 4.5a that for $\mu = 0.1$, FGM becomes better than GM after 50 iterations but for the other values of μ , the overrun is beyond 80 iterations and is therefore not visible on the plot.

Therefore, it is clear that the fast gradient method does not take advantage of the property of strong convexity unlike the gradient method.

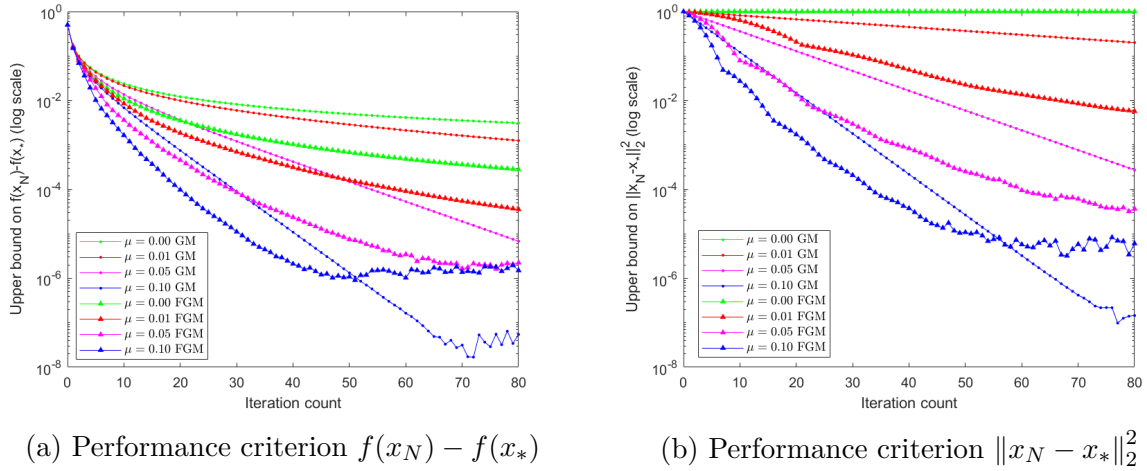


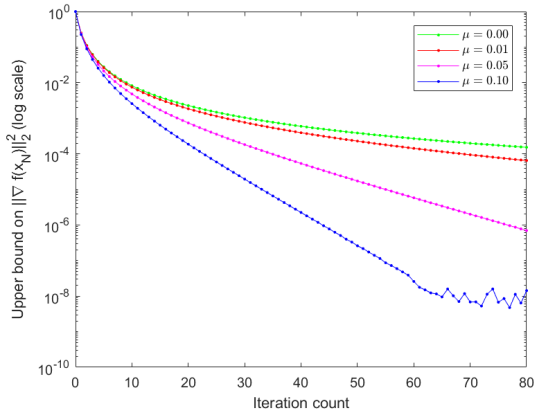
Figure 4.5: Comparison of the worst-case performance between GM and FGM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $f(x_N) - f(x_*)$ or $\|x_N - x_*\|_2^2$ as performance criterion.

4.4 Conclusion

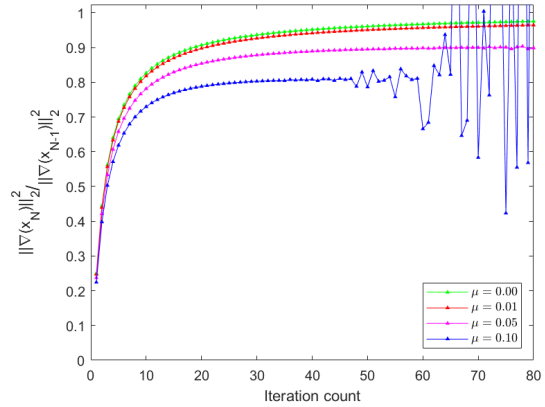
In this chapter, we have seen that, unlike the gradient method, the fast gradient method derives little benefits from the strong convexity property. Therefore, the goal of this master thesis is to improve the performance of this method for strongly convex functions. To this end, we will implement and analyze several versions of the fast gradient method.

4.A Worst-case performance and convergence for $\|\nabla f(x_n)\|_2^2$ as performance criterion

4.A.1 Worst-case performance and convergence of GM



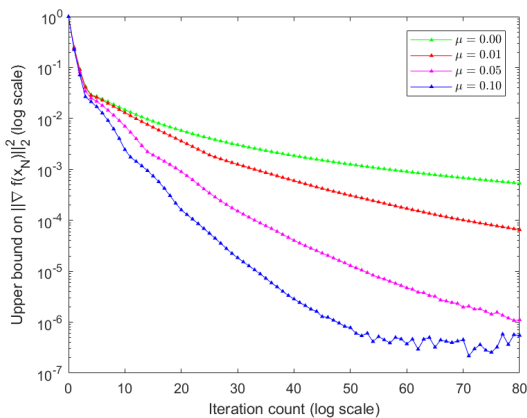
(a) Worst-case performance for several μ



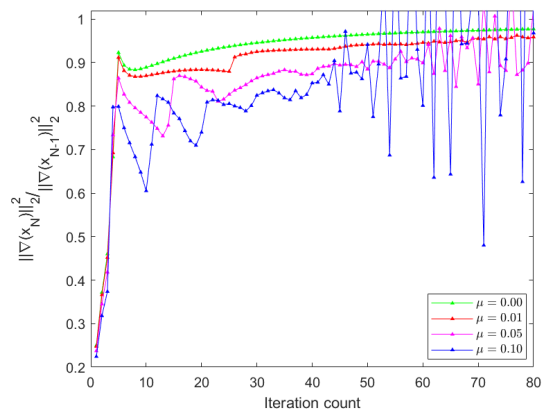
(b) Convergence for several μ

Figure 4.6: Comparison of the worst-case performance and the convergence of GM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $\|\nabla f(x_N)\|_2^2$ as performance criterion.

4.A.2 Worst-case performance and convergence of FGM



(a) Worst-case performance for several μ



(b) Convergence for several μ

Figure 4.7: Comparison of the worst-case performance and the convergence of FGM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $\|\nabla f(x_N)\|_2^2$ as performance criterion.

4.A.3 Comparison between GM and FGM

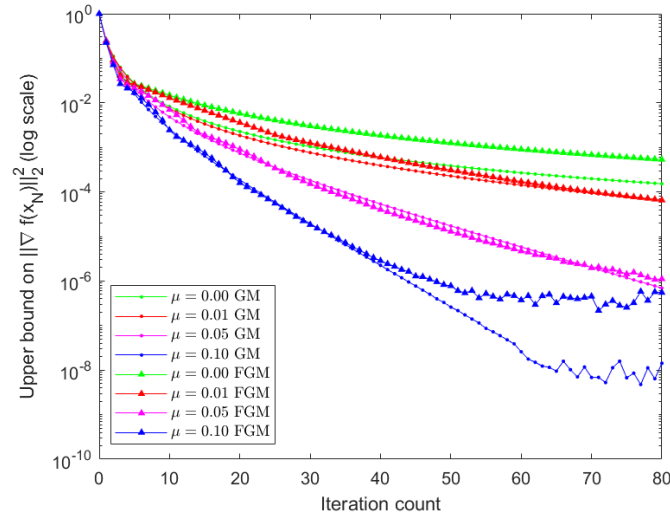


Figure 4.8: Comparison of the worst-case performance between GM and FGM for 1-smooth convex functions ($\mu = 0$) and for 1-smooth strongly convex functions ($\mu > 0$) with $\|\nabla f(x_N)\|_2^2$ as performance criterion.

In Chapter 4, we have seen that the goal of this master thesis is to improve the performance of the fast gradient method since this method does not exploit well the strong convexity property. The purpose of this chapter is therefore to provide a preamble on accelerated first-order methods introduced in the literature in the context of continuous unconstrained optimization. The implementation and the convergence rate of these schemes are detailed in the following.

5.1 Accelerated first-order methods

The accelerated first-order methods solving (Optimization Problem) were first developed by Yuri Nesterov in 1983 [Nes83]. He demonstrated that a slight modification of the gradient descent could obtain provably optimal performance for the class of first-order algorithms applied to minimize smooth convex functions. One of the most popular accelerated schemes is the fast gradient method that accelerates the gradient method using a momentum term $\beta_i(y_{i+1} - y_i)$ without additional computation. The general form of the fast gradient method is given by the following algorithm [KF15]:

FGM (general form)

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \alpha \nabla f(x_i)$$

$$x_{i+1} = y_{i+1} + \beta_i(y_{i+1} - y_i)$$

In this method, the parameters (α, β_i) must be properly selected to ensure a successful behavior of the algorithm. There are many accelerated schemes, e.g., in [CC19; BT09; KF15]. Table 5.1 summarizes some choices of the parameters and the convergence rates for the fast gradient methods FGM and FGM- q for smooth (possibly strongly) convex functions and the fast gradient method FGM- μ for smooth strongly convex functions. For these three methods, a step size of $\alpha = \frac{1}{L}$ is considered.

Table 5.1: Momentum coefficient convergence rates for several versions of the fast gradient method.

Method	Momentum coefficient β_i	Convergence rate
FGM	$\beta_i = \frac{\theta_i - 1}{\theta_{i+1}}, \begin{cases} \theta_{i+1} = \frac{1 + \sqrt{4\theta_i^2 + 1}}{2} \\ \theta_0 = 1 \end{cases}$	$f(x_N) - f(x_*) \leq \frac{2L\ x_0 - x_*\ _2^2}{(N+2)^2}$ [KF15]
FGM- q	$\beta_i = \frac{t_i(1-t_i)}{t_i^2 + t_{i+1}}, \begin{cases} t_{i+1}^2 = (1 - t_{i+1})t_i^2 + qt_{i+1} \\ t_0 = 1 \end{cases}, q \in [0, 1]$	$f(y_N) - f(x_*) \leq \frac{4L\ x_0 - x_*\ _2^2}{(N+2)^2}$ if $q = 0$ $f(y_N) - f(x_*) \leq L(1 - \sqrt{q})^N \ x_0 - x_*\ _2^2$ if $q = \frac{\mu}{L}$ [OC15]
FGM- μ	$\beta_i = \frac{1 - \sqrt{\frac{\mu}{L}}}{1 + \sqrt{\frac{\mu}{L}}} \quad \forall i$	$f(x_N) - f(x_*) \leq L(1 - \sqrt{\frac{\mu}{L}})^N \ x_0 - x_*\ _2^2$ [Nes04]

Note that, when $q = \frac{\mu}{L}$, the momentum coefficient β_i of FGM- q becomes closer to the momentum coefficient β_i of FGM- μ as i increases [OC15].

The convergence rate of FGM is referred to as sublinear while the convergence rate of FGM- μ is referred to as linear. Moreover, the FGM- q has also a linear convergence when $q = \frac{\mu}{L}$. Therefore, for smooth strongly convex functions where the strong convexity parameter is known in advance, the accelerated schemes FGM- μ and FGM- q improve the worst-case performance by enhancing the convergence rate.

However, these convergence rates are only guaranteed when the function parameters μ and L are known exactly. Therefore, methods for estimating these parameters have been implemented in the literature and some are presented below.

I Estimation of the Lipschitz constant

To estimate the Lipschitz constant, several efficient backtracking rules exist. Examples of these backtracking rules are available in [CC19; BT09; BCG10].

II Estimation of the strong convexity parameter

Estimating the strong convexity parameter is more complicated than estimating the Lipschitz constant [OC15]. To estimate this strong convexity parameter, Nesterov proposed a method to bound μ , which is similar to the backtracking scheme to estimate the Lipschitz constant [Nes13]. However, this scheme results in a slower convergence rate than FGM- μ and FGM- q .

In practice, many applications assume that μ is zero, which is equivalent to setting $q = 0$ in FGM- q . Indeed, many accelerated methods do not even include a q term, as FGM for example. But this choice significantly slows down the convergence of the iterates as can be seen in Table 5.1 for FGM- q .

Figure 5.1 shows the implementation of the accelerated scheme FGM- q on a positive definite quadratic function with the optimal choice of q being $q^* = \frac{\mu_*}{L_*} = 4.1 \times 10^{-5}$. Each line on the figure represents different values of $q = \frac{\mu}{L}$, i.e., we suppose that L_* is perfectly known and we test different estimates of the exact strong convexity

parameter μ^* . Over- or underestimating the optimal value q^* decreases the rate of convergence of the accelerated scheme. Moreover, the underestimation of q^* leads to the appearance of regular ripples while the overestimation of q^* leads to a monotonic convergence.

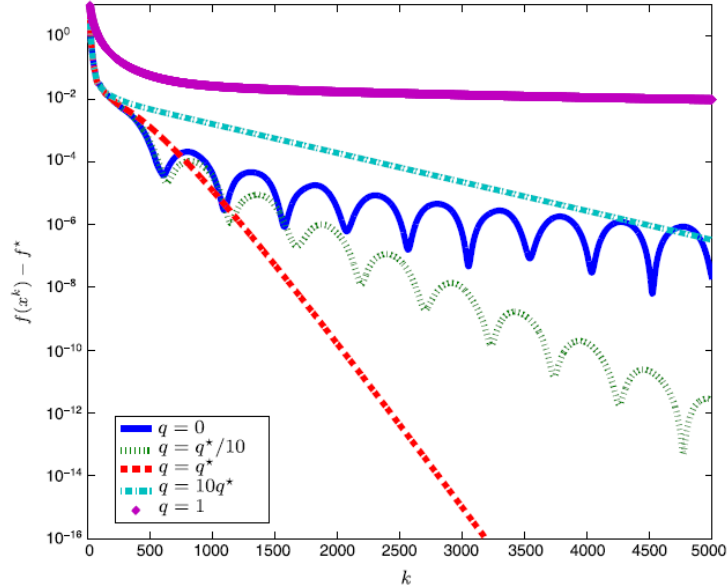


Figure 5.1: Convergence of FGM- q for several values of q . Reprinted with permission from [OC15].

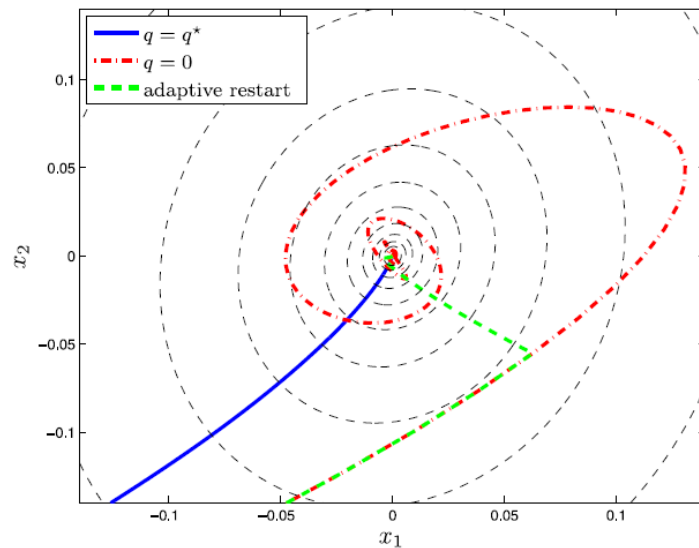


Figure 5.2: Sequence trajectories under FGM- q and with adaptive restart (see section 5.3). Reprinted with permission from [OC15].

Overestimating q^* is equivalent to underestimate the amount of momentum required since β_i decreases as q increases and thus, it leads to a slower convergence. However,

we overestimate more often the momentum coefficient because q is generally set to 0, in which case β_i tends to 1. This causes a rippling behavior in the objective function values (see Figure 5.1). To understand the appearance of these ripples, one can look in Figure 5.2 at the trajectories of the sequence generated by FGM- q minimizing a two-dimensional positive definite quadratic function and compare these trajectories for $q = q^*$, i.e., the optimal choice, and $q = 0$. We can observe that considering a high momentum coefficient, i.e., setting $q = 0$, leads to the trajectory overshooting the minimum and oscillating around it, which causes these ripples and significantly slows down the convergence.

Also, it is important to note that L and μ are global parameters and the function might be locally better conditioned in the region around the optimum. Outside this region, the optimal choice q^* is appropriate but inside this region, ripples are expected to emerge, despite the optimal choice of q .

To conclude, FGM- q or FGM- μ improve the performance of FGM for smooth strongly convex functions when the function parameters are known exactly. However, if the function is known to be strongly convex, tuning FGM- q or FGM- μ based on too conservative estimates of the strong convexity parameter leads to a slow down of the convergence and to a rippling behavior which makes the convergence even worse while tuning FGM- q or FGM- μ based on overestimates of the strong convexity parameter leads to monotonic but also slower convergence. Therefore, the accelerated schemes have been shown dramatically non-robust to the estimation of the μ parameter.

To address this, restart schemes have been introduced and studied in the literature: the fixed restart and the adaptive restart. These schemes are explained in the two following sections.

5.2 Fast gradient methods with fixed restart - FGM-FR

To improve the convergence of accelerated gradient schemes such as FGM, a fixed restart scheme has been introduced and studied in [Nem94; Nes13]. The following FGM-FR algorithm describes the procedure of the fast gradient method with fixed restart.

FGM-FR

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$ and restarting interval $k \geq 1$

For $i = 0, 1, \dots$ do

Running Algorithm (FGM) for k steps

Set $x_0 = x_k$, $y_0 = x_k$ and $\theta_0 = 1$

The idea of the fixed restart scheme is to restart the fast gradient method after a fixed number of iterations k , by taking as starting point the last point produced by the al-

gorithm. It means that all previous iterations are erased from the memory and the momentum coefficient β is reset to zero. Indeed, the momentum parameter θ is set to 1 at each restart and thus, $\beta_i = \frac{\theta_i - 1}{\theta_{i+1}} = 0$. To optimize the method and therefore have a better convergence, it is possible to determine the optimal restarting interval k based on the theoretical convergence rate as explained in the following subsection.

5.2.1 Optimal restarting interval k

Several authors such as [OC15; NNG16; KF15] have derived the optimal restarting interval, i.e., the optimal number of iterations between each restart, from theoretical upper bounds allowing to guarantee a global linear convergence for convex optimization problems.

Let us consider that we want to solve (Optimization Problem) for L -smooth μ -strongly convex functions using the fast gradient method with fixed restart. In other words, we restart FGM every k inner iterations by initializing the $(j+1)^{th}$ outer iteration as $x_{j+1,0} = x_{j,k}$ where $x_{j,i}$ is an iterate at the j^{th} outer iteration and i^{th} inner iteration. The goal is to find the optimal restarting interval k that minimizes the performance bounds from theorems in Subsection 2.3.2 for each performance criterion.

Optimal interval for $f(x_N) - f(x_*)$

Let us detail the computation of the optimal restarting interval, denoted k_* , with $f(x_N) - f(x_*)$ as performance criterion.

From Theorem 2.22, the performance bound for each outer iteration j is given by:

$$f(x_{j,k}) - f(x_*) \leq \frac{4L}{\mu(k+2)^2} (f(x_{j,0}) - f(x_*)) \leq \frac{4L}{\mu k^2} (f(x_{j,0}) - f(x_*)) \leq c(f(x_{j,0}) - f(x_*)), \quad (5.3)$$

where c is defined as $c = \frac{4L}{\mu k^2}$.

Then, considering that we apply j restarts every k iterations, the following bound is obtained:

$$\begin{aligned} f(x_{j,0}) - f(x_*) &= f(x_{j-1,k}) - f(x_*) && (x_{j+1,0} = x_{j,k}) \\ &\leq c(f(x_{j-1,0}) - f(x_*)) && (\text{inequality (5.3)}) \\ &= c(f(x_{j-2,k}) - f(x_*)) && (x_{j+1,0} = x_{j,k}) \\ &\leq c^2(f(x_{j-2,0}) - f(x_*)) && (\text{inequality (5.3)}) \\ &\vdots \\ &\leq c^j(f(x_{0,0}) - f(x_*)) = c^j(f(x_0) - f(x_*)). \end{aligned}$$

Afterwards, by taking $N = jk$ as the total number of iterations and denoting $x_N = x_{j,0}$, the above inequality becomes:

$$f(x_N) - f(x_*) \leq (c^{\frac{1}{k}})^N (f(x_0) - f(x_*)). \quad (5.4)$$

By minimizing the rate $(c^{\frac{1}{k}})^N$ according to the number of iterations k and using the monotonic logarithmic function to facilitate the minimization, as follows:

$$\min_k c^{\frac{1}{k}} \Leftrightarrow \min_k \frac{1}{k} \log c \Leftrightarrow \min_k \frac{1}{k} \log \frac{4L}{\mu k^2},$$

the following optimal restarting interval is obtained

$$k_* = 2e\sqrt{\frac{L}{\mu}} = 2e\sqrt{\kappa} \quad \text{and} \quad c = e^{-2},$$

where $e = 2.71828$ is the Euler's number and κ is the condition number $\kappa = \frac{L}{\mu}$.

Optimal interval for $\|x_N - x_*\|_2^2$

Then, let us find the optimal restarting interval for the performance criterion $\|x_N - x_*\|_2^2$. From Theorem [2.23](#), the performance bound is given by:

$$\|x_{j,k} - x_*\|_2^2 \leq \frac{4L}{\mu(k+2)^2} \|x_{j,0} - x_*\|_2^2 \leq \frac{4L}{\mu k^2} \|x_{j,0} - x_*\|_2^2,$$

and the constant $c = \frac{4L}{\mu k^2}$ is the same as for the performance criterion $f(x_N) - f(x_*)$. Therefore, the optimal restarting interval is the same for both performance criteria.

Optimal interval $\|\nabla f(x_N)\|_2^2$

Finally, from Theorem [2.24](#), the performance bound for the performance criterion $\|\nabla f(x_N)\|_2^2$ is given by:

$$\|\nabla f(x_{j,k})\|_2^2 \leq \frac{4L^2}{\mu^2(k+2)^2} \|\nabla f(x_{j,0})\|_2^2 \leq \frac{4L^2}{\mu^2 k^2} \|\nabla f(x_{j,0})\|_2^2,$$

and the constant $c = \frac{4L^2}{\mu^2 k^2}$. By applying a similar reasoning as the one for the two previous performance criteria, the optimal restarting interval can be determined as:

$$k_* = 2e\frac{L}{\mu} = 2e\kappa,$$

where $e = 2.71828$ is the Euler's number and κ is the condition number $\kappa = \frac{L}{\mu}$.

5.2.2 Convergence rate of FGM-FR

By replacing k by the optimal restarting interval k_* and c by e^{-2} in [\(5.4\)](#), we can bound the performance of [\(FGM-FR\)](#) with k_* as restarting interval:

$$f(x_N) - f(x_*) \leq (e^{-\frac{1}{e\sqrt{\kappa}}})^N (f(x_0) - f(x_*)) \quad \text{[Nes13]}.$$

An exponential can be expressed as $e^a = 1 + a$ when a is close to 0. Since $\frac{1}{e\sqrt{\kappa}}$ is close to zero, it leads to:

$$f(x_N) - f(x_*) \leq \left(e^{-\frac{1}{e\sqrt{\kappa}}}\right)^N (f(x_0) - f(x_*)) \approx \left(1 - \frac{1}{e\sqrt{\kappa}}\right)^N (f(x_0) - f(x_*)).$$

Therefore, to achieve a bound of $f(x_N) - f(x_*) \leq \epsilon$, at most $\mathcal{O}(e\sqrt{\kappa} \log(\frac{1}{\epsilon}))$ iterations of the fast gradient descent with fixed restart must be performed. The convergence rate of FGM-FR with $k = k_*$ is therefore **linear** since $\nu \in (0, 1)$ for the performance criterion $f(x_N) - f(x_*)$. A similar reasoning can be applied for the performance criteria $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$ and thus, we can show that the convergence rate of FGM-FR with $k = k_*$ is **linear** for all performance criteria considered.

In theory, the more the function is strongly convex, the more the optimal restarting interval k_* decreases. This seems logical because for FGM, the more μ increases, the faster the worst-case performance decreases at the beginning and then stagnates as seen in Chapter 4. Therefore, it is more optimal to apply the restart at the beginning of the process so that the worst-case performance continues to decrease rapidly.

However, the drawback of FGM-FR is that it requires prior knowledge of the condition number κ and particularly the strong convexity parameter μ which are not easily available in most practical cases. It would therefore be interesting to analyze the robustness of FGM-FR in order to determine whether the performance of FGM-FR is strongly impacted by a wrong estimation of the μ parameter. The robustness of FGM-FR will be analyzed in Chapter 7.

5.3 Fast gradient methods with adaptive restart - FGM-AR

To overcome the drawbacks of the fixed restart, Candès and O'Donoghue suggested another restarting strategy in [OC15] to improve the convergence of accelerated schemes by reducing the ripples in the function values: the adaptive restart scheme. This scheme consists of stopping the method when a certain restarting criterion on the objective function value or gradient is satisfied and then, restarting it using the last value provided by the algorithm as the new initial condition. One advantage of this scheme is that it does not require prior knowledge of μ . The procedure of the fast gradient method with adaptive restart FGM-AR is as follows:

FGM-AR

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$

For $i = 0, 1, \dots$ do

Running Algorithm (FGM)

If a condition is respected

Set $x_0 = x_k, y_0 = x_k$ and $\theta_0 = 1$

Candès and O’Donoghue were looking for conditions that decide whether or not to restart based on some computationally cheap observations. Two adaptive restart schemes (FS) and (GS) for FGM-AR have been proposed in [OC15]:

- Function scheme: restart whenever

$$f(y_{i+1}) > f(y_i). \quad (\text{FS})$$

To justify roughly this scheme, the adaptive restart is applied at the bottom of the troughs as in Figure 5.1, i.e., when the function value increases, which makes it possible to avoid wasted iterations where the iterate moves away from the optimum. Therefore, the function scheme allows reducing the rippling behavior.

- Gradient scheme: restart whenever

$$\langle \nabla f(x_i), y_{i+1} - y_i \rangle > 0. \quad (\text{GS})$$

The restart is applied when the momentum term seems to bring the iterate in a bad direction, i.e., when the momentum term and the negative gradient are making an obtuse angle.

The gradient scheme has two advantages over the function scheme. First, all quantities involved in the gradient scheme are already computed in the accelerated schemes, so no extra computation is required. Secondly, the gradient scheme is numerically more stable since $\nabla f(x_i)y_{i+1}$ tends to 0 when we approach the minimum while $f(y_i)$ gets closer to $f(x_*)$ which could cause possible cancellation error when evaluating $f(y_{i+1}) - f(y_i)$.

Then, Candès and O’Donoghue observed that, empirically, both schemes perform similarly well and improve the convergence performance of accelerated gradient schemes for smooth convex optimization. In [OC15], they have also provided a heuristic analysis to show that applying an adaptive restart allows recovering the optimal linear rate of convergence in many cases with no prior knowledge of function parameters.

Other restarting schemes have been implemented in the literature as in [GB14] where three restarting schemes (FS), (GMS) and (NMS) have been implemented :

- Gradient-mapping based scheme: restart whenever

$$\langle L(x_i - y_{i+1}), y_{i+1} - y_i \rangle > 0. \quad (\text{GMS})$$

- Non-monotonicity implying scheme: restart whenever

$$\langle L(x_{i-1} - y_i), y_{i+1} - \frac{1}{2}(y_i + x_{i-1}) \rangle > 0. \quad (\text{NMS})$$

When the restarting schemes described above and the (FS) scheme are applied to a lasso problem, they all perform similarly. More details are available in [GB14].

Therefore, it would be interesting to determine if the adaptive restart can improve the convergence of accelerated methods for the whole class of smooth (strongly) convex functions and not only in some particular cases. This is the focus of Chapter 8.

5.4 Conclusion

To conclude, in this chapter, we have seen that there exist several versions of FGM, such as FGM- q with $q = q_*$ or FGM- μ , that improve significantly the performance of FGM for smooth strongly convex functions when the function parameters are known exactly. However, under- or overestimating the momentum coefficient can have severe detrimental effects on the convergence of these accelerated gradient schemes. Therefore, tuning accelerated gradient methods designed for smooth strongly convex functions based on wrong estimates of μ leads to a significant slow down of the convergence: these methods are thus non-robust to wrong estimations of the parameter μ . Then, the restart scheme has been introduced to improve the convergence of accelerated gradient schemes such as the fast gradient method. More precisely, two strategies of restart have been introduced in the literature: the fixed restart and the adaptive restart.

On the one hand, the fixed restart consists of restarting the method after a fixed number of iterations k . For FGM-FR, it is possible to derive an optimal restarting interval k_* by minimizing the theoretical upper bounds on the different performance criteria. Using k_* requires the prior knowledge of the condition number and thus, of the strongly convex parameter which cannot be easily and precisely estimated in most practical cases.

On the other hand, the adaptive restart consists of restarting the method when a certain restarting criterion on the objective function value or gradient is satisfied. Several restarting criteria have been implemented in the literature and empirically, the adaptive restarting schemes seem to significantly improve the performance of accelerated gradient methods on specific problem instances without prior knowledge of function parameters.

From Chapter 4, we have seen that the general purpose of this master thesis is to improve the fast gradient method for strongly convex functions since the classical fast gradient method derives little benefits if the function turns out to be strongly convex. Moreover, from previous Chapter 5, we have concluded that we are looking for methods that do not depend on the strong convexity parameter μ or methods that are robust even when the estimation of μ is not accurate.

Several improvements of the fast gradient method are going to be proposed to enhance its performance for strongly convex functions. More precisely, we focus on improvement using a restart scheme. As mentioned in Chapter 5, restarting an algorithm means taking the current iterate as the new starting point after a certain number of iterations or when some restarting condition is satisfied. This erases the memory of previous iterations and resets the momentum parameter back to one.

To improve the convergence rate of fast gradient schemes for strongly convex functions, we focus on the implementation of three restarting procedures.

I **Fixed restart** - Chapter 7 will evaluate and analyze the performance of the fast gradient method with fixed restart, i.e., FGM-FR. The idea is to restart periodically the algorithm every k iterations, taking as a starting point the last point produced by the algorithm and resetting the momentum parameter θ to one. First, for each of the three performance criteria, the optimal restarting interval based on a specific criterion will be determined and compared to the theoretical restarting interval found in Chapter 5. From this optimal restarting interval, we will derive an optimal restarting interval formula. Then, the relative gain on the performance of FGM-FR depending on the choice of the restarting interval will be analyzed. Finally, we will evaluate and compare the robustness of FGM-FR using the optimal restarting interval formula and the version of the fast gradient method for strongly convex functions, i.e., FGM- μ , when the estimation of the strong convexity parameter μ is not accurate.

II **Adaptive restart** - Chapter 8 will try to implement and evaluate the performance of the fast gradient method with adaptive restart, i.e., FGM-AR. The idea is to restart the algorithm when a certain restarting condition on the objective function value or gradient is met. More precisely, we will consider two restarting conditions.

The first objective of this chapter will be to implement this method by dealing with the restarting condition which is strict and depends on the variables. Indeed, as explained in Chapter 3, PESTO cannot deal with the If conditions on variables neither with the strict conditions and therefore, some tricks will be required. Then, the worst-case performance of FGM-AR with the two restarting conditions will be analyzed. Finally, by considering a specific case of FGM-AR, we will show that the adaptive restart does not improve the performance of the classical fast gradient method.

- III **Soft restart** - Chapter 9 will implement and evaluate the performance of the fast gradient method with soft restart, i.e., FGM-SR. The idea is to apply a fraction of restart at each iteration by controlling the evolution of the momentum parameter θ instead of applying a full restart after a certain number of iterations. First, several strategies of soft restart will be described and analyzed. Secondly, the promising strategy(ies) will be explored more in detail for each performance criterion considered and we will show that one strategy improves the performance of FGM-FR. This strategy depends on a parameter called the restarting factor: we will determine the optimal restarting factor based on the same criterion as for the fixed restart. Then, from this optimal restarting factor, we will find an optimal restarting factor formula. Afterwards, the relative gain on the performance of FGM-SR using this strategy depending on the choice of the restarting factor will be analyzed. Finally, we will evaluate and compare the robustness of FGM-SR using the optimal restarting factor formula, FGM-FR using the optimal restarting interval formula and FGM- μ , i.e., the version of the fast gradient method for strongly convex functions when the estimation of the strong convexity parameter μ is not accurate.

As already mentioned in Chapter 4, in this master thesis, all numerical results are computed for the class of 1-smooth μ -strongly convex functions and with the initial condition $\|x_0 - x_*\|_2^2 \leq 1$. Therefore, the only parameter from the performance estimation problem that is going to be modified is μ .

7

Fast gradient method with fixed restart

This chapter first explains the fast gradient method with fixed restart and analyzes the performance of this method. Then, this chapter determines a strategy to find the optimal restarting interval and derives an accurate formula to estimate this optimal restarting interval for the criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. Finally, this chapter evaluates the gain of the method depending on the choice of the restarting interval and the robustness according to errors in the estimation of μ .

7.1 Method

As explained in Chapter 5, the idea of the fast gradient method with fixed restart is to restart periodically the fast gradient algorithm after a constant number of iterations k , by taking as starting point the last point produced by the algorithm and resetting the momentum parameter θ to one, i.e., setting $x_0 = x_k$, $y_0 = x_k$ and $\theta_0 = 1$ as described in Algorithm (FGM-FR). The fast gradient method with fixed restart is fully described in the following FGM-FR algorithm.

FGM-FR

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$, $\theta_0 = 1$ and restarting interval $k \geq 1$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$$

$$\theta_{i+1} = \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$$

$$x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}} (y_{i+1} - y_i)$$

If $(i + 1) \bmod k == 0$

$$\theta_{i+1} = 1$$

In practice, applying a fixed restart amounts to fixing the momentum parameter θ to 1 every k iterations. Indeed, $x_0 = x_k$ is equivalent to considering x_k to calculate y_{k+1} and when θ is set to 1, the momentum coefficient $\beta_k = \frac{\theta_k - 1}{\theta_{k+1}}$ is equal to 0 at the next iteration

and therefore, setting $y_0 = x_k$ is not necessary since it will be multiplied by $\beta = 0$.

Note that when the restarting interval k is set to 1, the fast gradient method with fixed restart corresponds actually to the gradient method since

$$k = 1 \Leftrightarrow \theta_{i+1} = 1 \ \forall i \Leftrightarrow \frac{\theta_i - 1}{\theta_{i+1}} = 0 \ \forall i \Leftrightarrow x_{i+1} = y_{i+1} \ \forall i \Leftrightarrow x_{i+1} = x_i - \frac{1}{L} \nabla f(x_i) \ \forall i.$$

Moreover, Chapter 5 has shown that accelerated schemes suffer from a rippling behavior in the objective function values when the momentum coefficient β is overestimated and this behavior significantly slows down the convergence of these schemes. Then, the restart has been introduced to improve their convergence. To understand how the restart impacts the convergence of the fast gradient method, let us analyze what happens after a restart at iteration i :

iteration i	$y_i = x_{i-1} - \frac{1}{L} \nabla f(x_{i-1})$ $x_i = y_i + \frac{\theta_{i-1}-1}{\theta_i} (y_i - y_{i-1})$ $\theta_i = 1$	Restart
iteration $i + 1$	$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$ $x_{i+1} = y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$	Usual gradient step
iteration $i + 2$	$y_{i+2} = y_{i+1} - \frac{1}{L} \nabla f(y_{i+1})$ $x_{i+2} = y_{i+2} + \frac{\theta_{i+1}-1}{\theta_{i+2}} (y_{i+2} - y_{i+1})$	Usual gradient step
iteration $i + 3$	$y_{i+3} = x_{i+2} - \frac{1}{L} \nabla f(x_{i+2})$ $x_{i+3} = y_{i+3} + \frac{\theta_{i+2}-1}{\theta_{i+3}} (y_{i+3} - y_{i+2})$ $\vdots$	

Note that usual gradient step means that a gradient step is applied from the same variable, i.e., $y_{i+1} = y_i - \frac{1}{L} \nabla f(y_i)$ or $x_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$.

We can see that one iteration after a restart (respectively two iterations after a restart), there is a usual gradient step and therefore, thanks to the Descent Lemma 2.12, we know that the function value in x (respectively in y) must decrease at this iteration. Thus, applying a restart can help to reduce the oscillatory behavior of accelerated schemes and to improve the convergence of FGM.

7.2 Worst-case performance analysis

In this section, we analyze the worst-case performance of FGM-FR using the toolbox PESTO for the three performance criteria considered in this master thesis. To visualize how FGM-FR behaves in a general way, we compute the performance of FGM-FR for several values of k and several values of μ . The number of iterations is limited to 80

since more iterations would require much more computational time, as explained in the drawbacks of Section 3.4.

In the following figures, the number of iterations between each restart, i.e., the restarting intervals, are chosen as $k \in \{5, 10, 20, 30, 60, 80, k_{the}\}$ where k_{the} corresponds to the theoretical restarting interval rounded to the nearest integer and is provided in the literature by minimizing the performance bounds, as explained in Section 5.2.1. More precisely, we have $k_{the} = \text{round}\left(2e\sqrt{\frac{L}{\mu}}\right)$ for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ and $k_{the} = \text{round}\left(2e\frac{L}{\mu}\right)$ for the performance criterion $\|\nabla f(x_N)\|_2^2$. Then, the case $k = 80$ corresponds to a classical FGM without restart since the total number of iteration is $N = 80$. Note that the theoretical restarting intervals for which the value of k_{the} is bigger than the total number of iterations $N = 80$ are not displayed in the figures. Indeed, in these cases, the algorithm will not have the opportunity to do a restart yet and the performance will be the same as the classical FGM.

First, in a general way, for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, we observe in Figures 7.1 and 7.2 that the larger the μ , the better the impact of the restart is and the faster the performance can be improved. Indeed, for small μ , applying a restart gives worse, i.e., bigger, performance than the classical FGM when the number of iterations N is limited to 80. But for large μ , some restarting intervals improve considerably the performance of FGM. For the performance criterion $\|\nabla f(x_N)\|_2^2$, Figure 7.3 shows that every restarting intervals k significantly improves the worst-case performance of FGM, even for small values of μ .

Therefore, the restart allows the fast gradient method to better exploit the strong convexity property for all performance criteria considered.

Afterwards, as mentioned in Chapter 4, numerical errors appear after a certain number of iterations when the worst-case performance is close to zero, i.e., less than 10^{-6} and then, the worst-case performance stagnates by oscillating around a certain value.

Moreover, in Figures 7.1, 7.2 and 7.3, we notice that applying a restart every k iterations produces similar rebounds of size k in the performance evolution for the criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ and it produces a similar pattern of size k in the performance evolution for the criterion $\|\nabla f(x_N)\|_2^2$ (without considering numerical errors). For the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, these rebounds are always decreasing while for the criterion $\|\nabla f(x_N)\|_2^2$, the restart allows decreasing sharply the performance at the beginning of the restart but then increases the performance until the next restart. For example, in Figure 7.3a for a restarting interval $k = 10$, the performance starts decreasing at the iteration 10 to then increasing from the iteration 14 until the next restart, i.e., until the iteration 20. Then, the same pattern starts again. Surprisingly, regardless of the value of the restarting interval k and the value of μ , the restart improves the worst-case performance only during 4 iterations and then worsens it until the next restart. However, even if it increases, the worst-case performance using a restart remains better than the one using no restart.

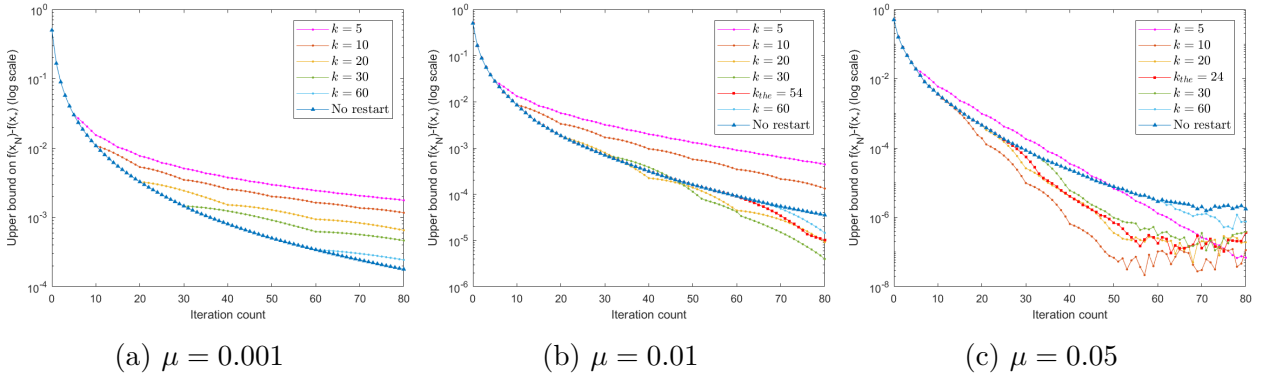


Figure 7.1: Comparison of the worst-case performance of FGM-FR for several restarting intervals k , for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

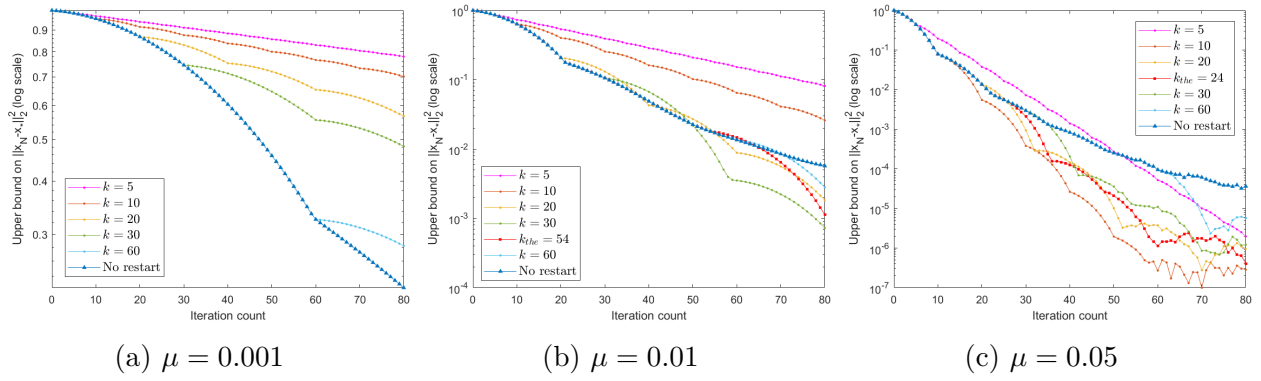


Figure 7.2: Comparison of the worst-case performance of FGM-FR for several restarting intervals k , for 1-smooth μ -strongly convex functions and $\|x_N - x_*\|_2^2$ as performance criterion.

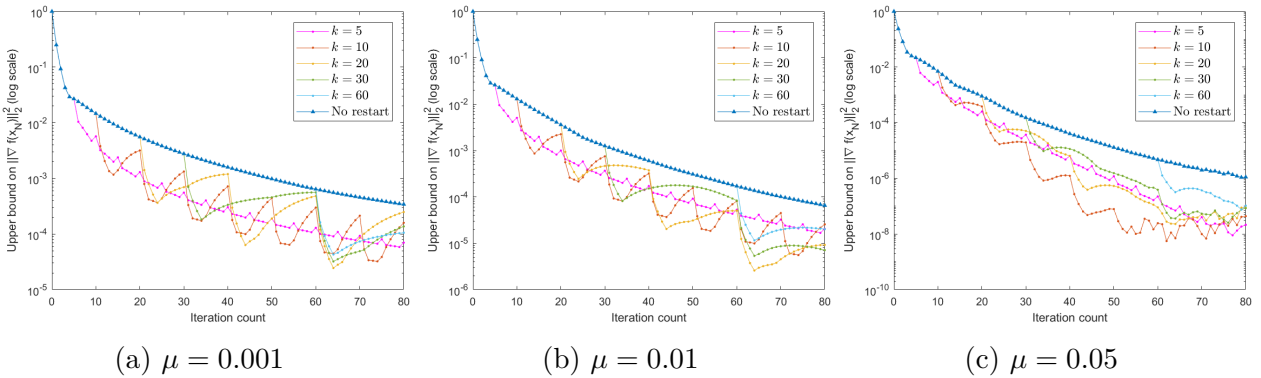


Figure 7.3: Comparison of the worst-case performance of FGM-FR for several restarting intervals k , for 1-smooth μ -strongly convex functions and $\|\nabla f(x_N)\|_2^2$ as performance criterion.

Next, we can see that, especially for high values of μ , the convergence rate seems to be linear for each restart interval k , for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. Indeed, even if some rebounds appear in the performance evolution for these performance criteria, the performance evolution could be interpolated by a straight line which confirms the linear convergence rate. For the performance $\|\nabla f(x_N)\|_2^2$, it is more subtle to evaluate the convergence rate but the performance evolution seems to go in the same direction as a linear shape, especially for high values of μ . Thus, this confirms that the convergence rate of the fast gradient method with fixed restart is linear, as shown in Chapter 5 for $k = k_{the}$.

Then, the optimal restarting interval k , i.e., the one giving the best worst-case performance, seems to be inversely proportional to μ for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. For example, for the criterion $\|x_N - x_*\|_2^2$, we can see on Figure 7.2b that the best restarting interval is around $k = 30$ for $\mu = 0.01$ and Figure 7.2c shows that the best restarting interval is around $k = 10$ for $\mu = 0.05$. Therefore, the optimal restarting interval k seems to increase as μ decreases. For the criterion $\|\nabla f(x_N)\|_2^2$, determining the optimal restarting interval k is more complicated because the worst-case performance oscillates a lot by decreasing and increasing after each restart. Indeed, in Figure 7.3c, we can observe that the best restarting interval seems to be $k = 10$ for $\mu = 0.05$ but it is much more difficult to determine which restarting interval is optimal for $\mu = 0.001$ on Figure 7.3a. To determine objectively the optimal restarting interval, a specific criterion will be defined in the next section.

Finally, it also appears that the theoretical restarting interval k_{the} found in the literature does not correspond to the optimal restarting interval for the three performance criteria considered. For example, Figure 7.1c shows that for $\mu = 0.05$, the optimal restarting interval seems to be close to 10 rather than close to $k_{the} = 24$. For the performance criterion $\|\nabla f(x_N)\|_2^2$, Figure 7.3 does not even display the worst-case performance for the restarting interval k_{the} since k_{the} is greater than 80 for all values of μ considered: k_{the} is equal to 103, 542 and 5437 respectively for μ equal to 0.05, 0.01 and 0.001. This means that the theoretical performance bounds from Theorems 2.22, 2.23 and 2.24 are not tight, especially for the performance criterion $\|\nabla f(x_N)\|_2^2$, since these upper bounds are used to determine k_{the} which does not seem to be the optimal restarting interval. Therefore, to determine the «true» optimal restarting interval k_{opt} , the next section implements an automatic strategy based on an objective criterion and try to determine a more accurate formula.

7.3 Optimal restarting interval

In this section, a strategy based on a specific criterion is explained and implemented in order to determine the optimal restarting interval. Then, the optimal restarting interval is compared to the theoretical restarting interval found in the literature.

7.3.1 Strategy

To find a more accurate formula than the theoretical restarting interval k_{the} , we need to define a criterion that allows us determining objectively the best restarting interval for each value of μ . The criterion chosen to identify the best restarting interval is the following one.

Definition 7.1 (Criterion $N_{\mu,\epsilon}(k)$). The *criterion* $N_{\mu,\epsilon}(k)$ is the smallest number of iterations needed for FGM-FR with the restarting interval k to reach a specific precision ϵ on the worst-case performance for 1-smooth μ -strongly convex functions.

Then, the best restarting interval k_{opt} corresponds to the restarting interval k that minimizes $N_{\mu,\epsilon}(k)$ for a specific value of μ and a precision ϵ :

$$k_{opt}(\mu, \epsilon) = \arg \min_k N_{\mu,\epsilon}(k).$$

In practice, to compute this criterion for given values of μ , ϵ and k and for a specific performance criterion, the idea is to evaluate the worst-case performance of FGM-FR for 1-smooth μ -strongly convex functions with the restarting interval k for every number of iterations from 1 until the performance is less than or equal to ϵ : the last number of iterations considered corresponds then to $N_{\mu,\epsilon}(k)$.

Moreover, to save time, for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ which decrease monotonically according to N for FGM-FR, we compute the precision reached by FGM-FR every 10 iterations until the desired precision is reached. Therefore, the exact number of iterations $N_{\mu,\epsilon}$ is either this iteration or one of the nine previous ones and to find it, we inspect every iteration in this interval until the desired accuracy is reached.

Afterwards, to determine the optimal restarting interval k_{opt} , we have applied the following strategy based on three points: the choice of a precision ϵ , the choice of an initial searching space for the values of k and the limitation of the calculation of $N_{\mu,\epsilon}(k)$ to a maximum number of iterations.

I Precision - We limit ourselves to a minimum precision of $\epsilon_{min} = 10^{-5}$ since numerical errors appear from a precision of 10^{-6} and the precision ϵ is adjusted according to the value of μ . Indeed, as previously mentioned in Section 3.4, since we restrict ourselves to a maximum number of iterations $N_{max} = 200$ ¹, according to the value of μ and the performance criterion, the performance may not reach a precision of 10^{-5} after 200 iterations. Therefore, for the two performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, to determine the precision, we calculate the worst-case performance of FGM-FR with the theoretical restarting interval k_{the} after N_{max} iterations for 1-smooth μ -strongly convex functions. If the computed performance is lower than 10^{-5} , then the precision is set as $\epsilon = \epsilon_{min} = 10^{-5}$. Otherwise, the precision ϵ is set as the order of precision of the worst-case performance calculated for the theoretical restarting interval k_{the} .

¹We were able to solve problems for 200 iterations using the server but after an update, memory problems appeared and therefore, we considered $N_{max} = 150$ after this update. The results obtained with this N_{max} are marked in red in the following tables.

For the performance criterion $\|\nabla f(x_N)\|_2^2$, since k_{the} is too high, the performance of the classical FGM reaches the precision $\epsilon = 10^{-5}$ for a number of iterations $N < k_{the}$ and therefore, no restart is triggered. Thus, we do not consider k_{the} for this performance criterion and the precision is simply set to 10^{-5} .

The following table resumes the precision thresholds according to the different values of μ and according to the performance criteria. For example, we can observe that for the performance criterion $\|x_N - x_*\|_2^2$, a lower precision is considered for small μ , i.e., $\mu < 0.01$, since for these values of μ , the worst-case performance does not reach an accuracy smaller than 10^{-5} after N_{max} iterations.

Table 7.1: Precision according to the value of μ and the performance criterion^[1].

	$\ x_N - x_*\ _2^2$	$f(x_N) - f(x_*)$	$\ \nabla f(x_N)\ _2^2$
$\mu \in [0.2, 0.01]$	10^{-5}	10^{-5}	10^{-5}
$\mu = 0.007$	$2 \cdot 10^{-5}$	10^{-5}	10^{-5}
$\mu = 0.005$	$5 \cdot 10^{-4}$	10^{-5}	10^{-5}
$\mu = 0.002$	$2 \cdot 10^{-2}$	10^{-5}	10^{-5}
$\mu \in [0.001, 0.0001]$	/	/	10^{-5}

II Searching space - The goal is to find the restarting interval k that minimizes $N_{\mu,\epsilon}(k)$. However, it would require too much computational time to compute the criterion $N_{\mu,\epsilon}(k)$ for every possible restarting interval k . Indeed, to determine $N_{\mu,\epsilon}(k)$ for one value of k , we need to compute the worst-case performance for every number of iterations $N \leq N_{\mu,\epsilon}(k)$ which takes a lot of time, especially when the criterion $N_{\mu,\epsilon}(k)$ is high (see Section 3.4).

Therefore, the strategy is to determine an initial searching space K depending on the values of μ and ϵ which corresponds to the first values of restarting intervals k to be tested and then, we refine our search around the local minima. For a given μ , the highest restarting interval k_{max} considered is the minimum between the maximum number of iterations, i.e., $N_{max} = 200$ ^[1], and the number of iterations needed to reach a precision of ϵ for FGM, i.e., $N_{\mu,\epsilon}(200)$. Indeed, if, for example, the classical FGM requires 20 iterations to reach an accuracy of ϵ , there is no need to test a restarting interval of 25 since the accuracy will be reached before the restart is triggered. Therefore, we have

$$k_{max}(\mu, \epsilon) = \min(200, N_{\mu,\epsilon}(200)).$$

Then, when k_{max} is small, we can consider a small interval between each value of k in the initial searching space by keeping a computational time relatively low. As k_{max} increases, we consider a larger interval between each value of k in the initial searching space. The searching space according to the value of k_{max} is described in Table 7.2. Note that the smallest restarting interval considered is 2 since the restarting interval equal to 1 corresponds to the gradient method, as explained above in Section 7.1.

Table 7.2: Initial searching space K according to the value of k_{max} .

	Searching space K
$1 \leq k_{max} \leq 60$	$K = \{2, 5, 10, 15, 20, 25, \dots, k_{max}\}$
$60 < k_{max} \leq 100$	$K = \{10, 20, 30, 40, 50, \dots, k_{max}\}$
$100 < k_{max} \leq 200$	$K = \{10, 20, 40, 60, 80, \dots, k_{max}\}$

Afterwards, once the criterion $N_{\mu,\epsilon}(k)$ is computed for all $k \in K$, we focus on the most promising local minima and refine our search around them. A restarting interval $k_i \in K$ is a local minimum if $N_{\mu,\epsilon}(k_{i-1}) \geq N_{\mu,\epsilon}(k_i)$ and $N_{\mu,\epsilon}(k_i) \leq N_{\mu,\epsilon}(k_{i+1})$. More precisely, we consider the smallest local minimum found and other local minima whose values are at most 10% higher than the smallest local minimum if there are any. In the further explanation, the set of considered local minima, i.e., the smallest one and the 10% higher ones, is denoted as K_* and is given by:

$$K_* = \{k \text{ such that } N_{\mu,\epsilon}(k) \leq 1.1 \cdot \min_k N_{\mu,\epsilon}(k) \quad \forall k \in K\}.$$

Then, we refine our search by computing the criterion $N_{\mu,\epsilon}$ for every 10 restarting intervals around the $k \in K_*$, i.e., for $\{k-10, k+10\}$ with $k \in K_*$, and K_* is updated. After, we compute the criterion for every 5 restarting intervals around the $k \in K_*$, i.e., for $\{k-5, k+5\}$ with $k \in K_*$, and K_* is updated again. Finally, we compute the criterion for every restarting interval around the $k \in K_*$, i.e., for $\{k-4, \dots, k-1, k+1, \dots, k+4\}$ with $k \in K_*$, and we can determine k_{opt} . In the case where the number of iterations $N_{\mu,\epsilon}$ for one restarting interval is already computed in the initial searching space, the result is stored to avoid recalculating it again. For example, for a value of $60 < k_{max} \leq 100$, the criterion $N_{\mu,\epsilon}$ is computed for every k in the initial searching space K and then, for every 5 restarting intervals around all k in the set K_* (and not for every 10 restarting intervals since these are already computed in the initial searching space K) and finally, the criterion is computed for every restarting interval around the new set K_* .

- III **Limited number of iterations** - Since the goal is to find the smallest criterion $N_{\mu,\epsilon}(k)$, for a given k , we can stop the computation as soon as the current minimum number of iterations has been exceeded. Therefore, we stop the calculation of the number of iterations after a maximum number of iterations N_{lim} such that

$$N_{lim} = \min(N_{max}, \lceil 1.15 \cdot N_{min} \rceil),$$

where N_{min} is the current smallest number of iterations needed to reach an ϵ precision and is initialized to $N_{max} = 200$ [1]. The factor 1.15 allows to detect any potential promising local minimum.

7.3.1.1 Example

In order to have a good understanding of the strategy, let us apply it to an example: find the optimal restarting interval k_{opt} of FGM-FR for 1-smooth 0.07-strongly convex

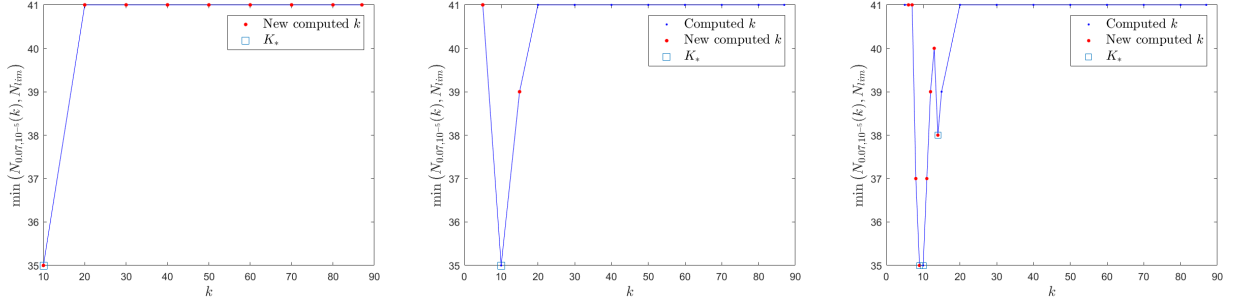
functions with the performance criteria $\|x_N - x_*\|_2^2$. Figure 7.4 displays the several steps applied.

First, we need to determine the **precision** threshold for $\mu = 0.07$ and thus, we first calculate the worst-case performance of FGM-FR with the theoretical restarting interval $k_{the} = \text{round}\left(2e\sqrt{\frac{L}{\mu}}\right) = 21$ at the maximum number of iterations $N_{max} = 200$. Since this performance is lower than $\epsilon_{min} = 10^{-5}$, the precision threshold is set to $\epsilon = 10^{-5}$.

Then, to determine the initial **searching space**, we need to compute the highest restarting interval considered k_{max} as follows:

$$k_{max}(0.07, 10^{-5}) = \min(200, N_{0.07, 10^{-5}}(200)) = \min(200, 87) = 87.$$

Therefore, the initial search space is defined as $K = \{10, 20, 30, 40, 50, \dots, 80, 87\}$ (see Table 7.2).



(a) Step 1: exploration in the initial searching space $K = \{10, 20, \dots, 80, 87\}$. The set of local minima is $K_* = 10$.

(b) Step 2: search every 5 restarting intervals around the local minimum $k = 10$, i.e., compute the criterion for $k \in \{5, 15\}$. The set of local minima is $K_* = 10$.

(c) Step 3: search every restarting interval around the local minimum $k = 10$, i.e., compute the criterion for $k \in \{6, 7, 8, 9, 11, 12, 13, 14\}$. The optimal k is $k_{opt} \in \{9, 10\}$.

Figure 7.4: Implementation of the strategy to determine the optimal restarting interval(s) k_{opt} for 1-smooth 0.07-strongly convex functions on the performance criterion $\|x_N - x_*\|_2^2$ and using a precision threshold of $\epsilon = 10^{-5}$.

Afterwards, the first step computes the criterion for every restarting interval $k \in K$ as observed in Figure 7.4a, where the red points correspond to the searching points. Then, the first local minimum is found and the set K_* is initialized as $K_* = \{10\}$.

Note that, as explained previously, the calculation of the number of iterations is stopped when the **limited number of iterations** N_{lim} is reached. In our example, after the computation of the first k in K , i.e., for $k = 10$, we have:

$$N_{lim} = \min(200, \lceil 1.15 \cdot N_{min} \rceil) = \min(200, \lceil 1.15 \cdot N_{0.07, 10^{-5}}(10) \rceil) = 41.$$

In the second step, the search is restricted to every 5 restarting intervals around the restarting interval k in the set $K_* = \{10\}$. Therefore, we compute the criterion for $k = 5$ and $k = 15$ as illustrated in Figure 7.4b. The set of local minima K_* remains the same.

The third step searches every restarting interval around the local minima belonging to the set K_* , i.e., around $k = 10$, and thus, we compute the criterion for every restarting interval $k \in \{6, 7, 8, 9, 11, 12, 13, 14\}$. As observed in Figure 7.4c, the search detects two more local minima: $K_* = \{9, 10, 14\}$. Since we have already searched around these local minima, we can conclude that the optimal restarting interval of FGM-FR for 1-smooth 0.07-strongly convex functions with the performance criteria $\|x_N - x_*\|_2^2$ is given by $k_{opt} \in \{9, 10\}$.

7.3.2 Analytical results

The above strategy is applied for all performance criteria considered and for different values of μ from 0.002 to 0.2 for the performance criteria $\|x_N - x_*\|_2^2$ and $f(x_N) - f(x_*)$ and from 0.0001 to 0.2 for the performance criterion $\|\nabla f(x_N)\|_2^2$.

On the one hand, the bigger the value of μ the better the function is already well-conditioned and therefore, FGM already works very well and converges fast, even without restart. This is why we do not consider μ bigger than 0.2.

On the other hand, the smallest value of μ considered is 0.002 for the performance criteria $\|x_N - x_*\|_2^2$ and $f(x_N) - f(x_*)$. Indeed, we want to compare the optimal restarting interval with the theoretical one and since k_{the} is too high for smaller values of μ , it would give same results than k_{max} : $k_{the} = 172 > N_{max} = 150$ for $\mu = 0.001$ [1]. For the performance criterion $\|\nabla f(x_N)\|_2^2$, we do not consider k_{the} because it is too high and thus, the smallest value of μ considered is 0.0001.

The following tables show the results for all performance criteria and several values of μ .

Table 7.3: Comparison between the optimal restarting interval k_{opt} and the theoretical restarting interval k_{the} for the performance criteria $\|x_N - x_*\|_2^2$ and $f(x_N) - f(x_*)$ and several values of μ [1].

μ	k_{the}	k_{opt} $\ x_N - x_*\ _2^2$	k_{opt} $f(x_N) - f(x_*)$
0.2	12	5	5
0.17	13	5-6	5
0.15	14	6	5-6
0.12	16	7	6-7
0.1	17	7-8	7
0.07	21	9-10	9
0.05	24	11-12	10
0.02	38	19-20	17
0.01	54	27-28	24-27
0.007	65	32	28
0.005	77	37-38	33-39
0.002	122	58-59	61

First, Table 7.3 shows that the optimal restarting interval k_{opt} is inversely proportional to μ for both performance criteria, which confirms the observations from Section 7.2.

Indeed, the lower the value of μ , the bigger the optimal restarting interval is. Then, we can see that the optimal restarting interval k_{opt} follows the same evolution for both performance criteria. This is rather interesting since it means that choosing k as k_{opt} for $f(x_N) - f(x_*)$ also minimizes the number of iterations needed to reach a certain accuracy for the performance criterion $\|x_N - x_*\|_2^2$ and inversely. Another observation is that the theoretical restarting formula $k_{the} = \text{round}\left(2e\sqrt{\frac{L}{\mu}}\right)$ does not seem to fit with the values of the optimal restarting intervals k_{opt} : by analyzing the values in the table, it seems that for both performance criteria, k_{the} is more or less the double of k_{opt} .

Then, Table 7.4 confirms that the theoretical restarting formula k_{the} found in the literature for the performance criterion $\|\nabla f(x_N)\|_2^2$ does not match at all with the optimal values k_{opt} . Then, we observe that the optimal restarting interval starts to stagnate for small values of μ , i.e., for $\mu \leq 0.01$. These results could be explained by the fact that for this performance criterion, we have seen in Section 7.2 that all the restarting intervals k help to improve significantly the performance of FGM for all values of μ . Therefore, it is better to apply a restart fast, i.e., to consider a relatively small restarting interval k , in order to quickly improve the performance. Thus, the optimal restarting interval for the performance criterion $\|\nabla f(x_N)\|_2^2$ does not appear to be inversely proportional to μ and seems to converge to a range of values around 19.

Table 7.4: Comparison between the optimal restarting interval k_{opt} and the theoretical restarting interval k_{the} for the performance criterion $\|\nabla f(x_N)\|_2^2$.

μ	k_{the}	k_{opt} $\ \nabla f(x_N)\ _2^2$
0.2	27	4-5
0.17	32	5
0.15	36	6
0.12	45	7
0.10	54	6-8
0.07	78	7
0.05	109	8
0.02	272	12
0.01	544	16
0.007	777	19
0.005	1087	21
0.002	2718	16-20
0.001	5437	18
0.0005	10873	19
0.0001	54366	17

To conclude, we observe that globally, for all the performance criteria, the optimal restarting intervals do not match with the theoretical restarting intervals: indeed, the theoretical optimal restarting intervals are greater than the empirical optimal ones. This means that the upper bounds on the performance found in the literature are not tight and better approximations could be derived to determine the restarting interval k .

7.4 Optimal restarting interval formula

In Subsection 7.3.2, we observed that the optimal restarting interval k_{opt} does not correspond to the theoretical one found in the literature. Therefore, it would be relevant to determine an optimal formula that better fits the optimal restarting intervals found in Tables 7.3 and 7.4 for each performance criterion.

For the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, Table 7.3 shows that there is a ratio of approximately two between the restarting intervals k_{the} and k_{opt} . Thus, we wanted to explore if the formula $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$ matched the values of optimal restarting intervals k_{opt} . Figure 7.5 displays the formula $\hat{k}_{opt}$ and compares it to the optimal restarting interval k_{opt} and the theoretical restarting interval $k_{the} = \text{round}\left(2e\sqrt{\frac{L}{\mu}}\right)$. As expected, $\hat{k}_{opt}$ seems to fit with k_{opt} for a large range of values of μ . Therefore, the formula to approximate the optimal restarting interval is $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$ for both performance criteria $f(x_N) - f(x_*)$ or $\|x_N - x_*\|_2^2$.

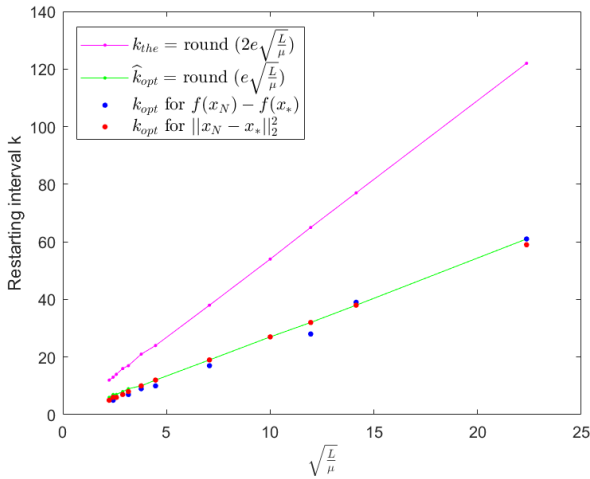


Figure 7.5: Comparison between the optimal restarting intervals k_{opt} , the theoretical restarting interval formula k_{the} and the optimal restarting interval formula $\hat{k}_{opt}$ for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$.

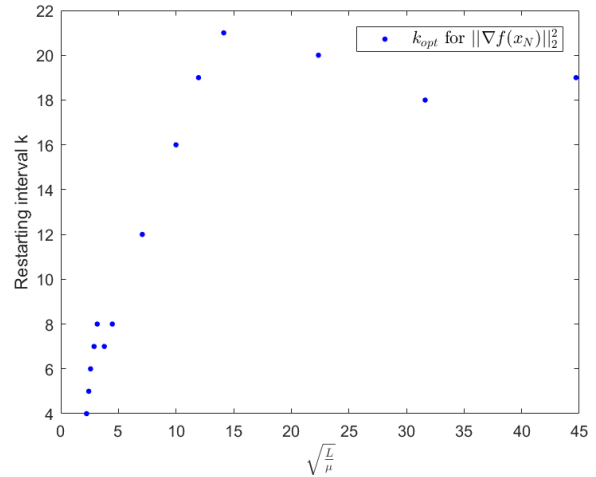


Figure 7.6: Optimal restarting intervals k_{opt} for the performance criterion $\|\nabla f(x_N)\|_2^2$.

For $\|\nabla f(x_N)\|_2^2$, Figure 7.6 displays the optimal restarting intervals k_{opt} resulting from Strategy 7.4. We can observe that k_{opt} evolves according to several regimes: for large values of μ , i.e., when $\sqrt{\frac{L}{\mu}}$ is small, k_{opt} increases rapidly as a function of $\sqrt{\frac{L}{\mu}}$, then more slowly and finally stagnates when $\sqrt{\frac{L}{\mu}} \geq 15$. Moreover, even if we find a formula to approximate the three regimes, the k obtained with this formula could not perform well if they do not fit perfectly with k_{opt} . Indeed, since the performance of FGM-FR oscillates a lot by decreasing and then increasing after each restart as seen in Section 7.2, even if

the criterion $N_{\mu,\epsilon}(k)$ is small for a restarting interval k , it could be much higher for the restarting interval $k+1$. So, the performance of FGM-FR with $k+1$ as restarting interval could increase before reaching an ϵ precision and we could have $N_{\mu,\epsilon}(k) \ll N_{\mu,\epsilon}(k+1)$. Therefore, finding a formula for the performance criterion $\|\nabla f(x_N)\|_2^2$ that gives good results is complicated and we did not find a formula.

Finally, Table 7.5 compares the theoretical restarting formula and the optimal restarting interval formula for each performance criterion.

Table 7.5: Comparison between the optimal restarting interval formula $\hat{k}_{opt}$ and the theoretical restarting formula k_{the} for each performance criterion.

Performance criterion	$f(x_N) - f(x_*)$	$\ x_N - x_*\ _2^2$	$\ \nabla f(x_N)\ _2^2$
k_{the}	$\text{round}\left(2e\sqrt{\frac{L}{\mu}}\right)$	$\text{round}\left(2e\sqrt{\frac{L}{\mu}}\right)$	$\text{round}\left(2e\frac{L}{\mu}\right)$
$\hat{k}_{opt}$	$\text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$	$\text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$	$\times$

7.5 Gain

To determine whether the fast gradient method with fixed restart allows to really improve the worst-case performance of the classical FGM, we compute and compare the criterion $N_{\mu,\epsilon}(k)$ for $k \in \{k_{opt}, \hat{k}_{opt}, k_{the}, k_{max}\}$, for the same values of μ and for each performance criterion. Note that $k = k_{max}$ corresponds to the classical FGM since no restart is applied for this restarting interval.

To this end, the precision is adjusted according to the value of μ and the performance criterion as in Table 7.1. Therefore, for the performance criterion $\|x_N - x_*\|_2^2$, the criterion $N_{\mu,\epsilon}(k)$ for the smallest values of μ , i.e., $\mu \leq 0.007$, is calculated for a higher precision than 10^{-5} while for the other performance criteria, $\epsilon = 10^{-5}$ for all values of μ is considered.

Tables 7.6, 7.7, 7.8 display the results and Figures 7.7, 7.8, 7.9 plot the criterion $N_{\mu,\epsilon}(k)$ only for the same precision $\epsilon = 10^{-5}$ in order to display comparable results. Thus, for the performance criterion $\|x_N - x_*\|_2^2$, Figure 7.8 displays the results only for $\mu \geq 0.01$. Moreover, since the maximum number of iterations is $N_{max} = 200$ (or $N_{max} = 150$ after the update of the server), $N_{\mu,\epsilon}(k)$ can be bigger than N_{max} for some values of μ when $k = k_{max}$. Therefore, on Figures 7.7, 7.8 and 7.9, the unfilled points correspond to a number of iterations strictly greater than N_{max} .

Then, we also analyze the relative gain depending on the choice of the restarting intervals $k = \{k_{opt}, \hat{k}_{opt}, k_{the}, k_{max}\}$. The gain is calculated as the difference between the number of iterations required to reach some precision for two restarting intervals. Therefore, the relative gain is defined as follows:

$$G_{\mu,\epsilon}(k_1, k_2) = \frac{N_{\mu,\epsilon}(k_2) - N_{\mu,\epsilon}(k_1)}{N_{\mu,\epsilon}(k_2)}.$$

The relative gain corresponds to the percentage of iterations we gain to reach a certain precision if we use k_1 instead of k_2 . Note that this formula cannot be used when $N_{\mu,\epsilon}(k) > N_{max}$ but only when $N_{\mu,\epsilon}(k)$ is known exactly.

Table 7.6: Comparison of $N_{\mu,10^{-5}}(k)$ for different values of μ and k with $f(x_N) - f(x_*)$ as performance criterion.

μ	k_{opt}	$\hat{k}_{opt}$	k_{the}	k_{max}
0.2	13	14	16	19
0.17	14	17	18	22
0.15	16	17	19	24
0.12	18	20	22	28
0.1	20	22	24	31
0.07	25	26	30	39
0.05	30	32	36	48
0.02	50	51	58	80
0.01	71	71	81	116
0.007	84	85	95	135
0.005	98	99	109	156
0.002	143	143	167	198

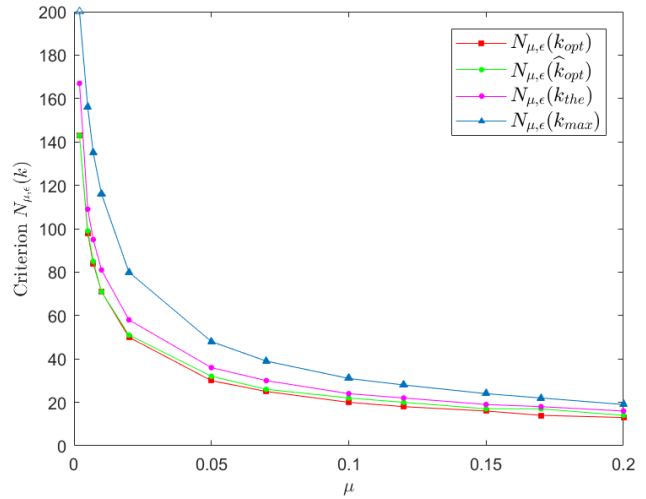


Figure 7.7: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ according to μ for $k \in \{k_{opt}, \hat{k}_{opt}, k_{max}, k_{the}\}$ with $f(x_N) - f(x_*)$ as the performance criterion.

Table 7.7: Comparison of $N_{\mu,\epsilon}(k)$ for different values of μ and k with $\|x_N - x_*\|_2^2$ as performance criterion^[1].

μ	k_{opt}	$\hat{k}_{opt}$	k_{the}	k_{max}
0.2	15	17	20	25
0.17	18	19	23	30
0.15	19	20	25	34
0.12	23	24	30	40
0.1	27	29	33	53
0.07	35	35	49	87
0.05	43	44	54	> 200
0.02	76	76	92	> 200
0.01	122	122	172	> 200
0.007	136	142	196	> 200
0.005	116	124	147	> 150
0.002	116	118	150	> 150

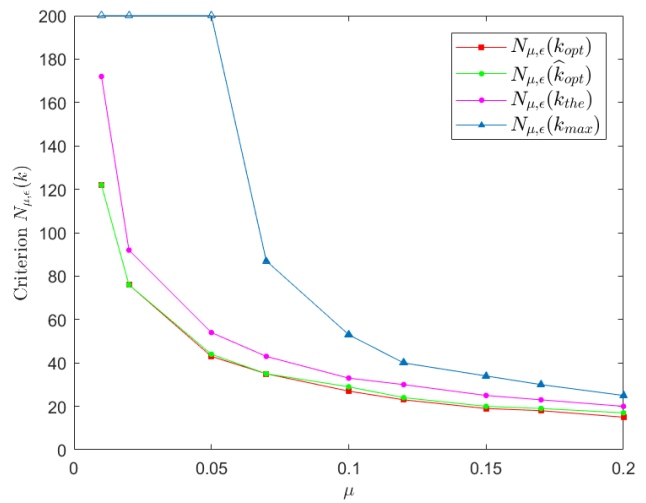


Figure 7.8: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ according to μ for $k \in \{k_{opt}, \hat{k}_{opt}, k_{max}, k_{the}\}$ with $\|x_N - x_*\|_2^2$ as the performance criterion.

Table 7.8: Comparison of $N_{\mu,10^{-5}}(k)$ for different values of μ and k with $\|\nabla(x_N)\|_2^2$ as performance criterion.

μ	k_{opt}	k_{the} or k_{max}
0.2	13	21
0.17	13	23
0.15	15	25
0.12	17	30
0.1	20	34
0.07	23	43
0.05	27	53
0.02	40	92
0.01	52	134
0.007	61	163
0.005	67	193
0.002	84	>200
0.001	94	>200
0.0005	99	>200
0.0001	106	>200

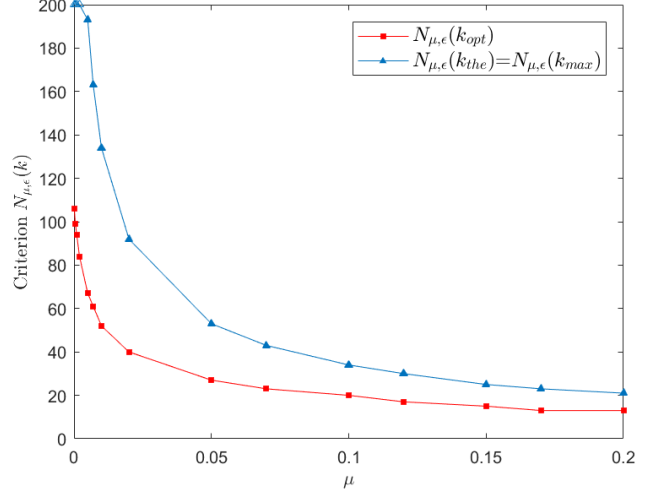


Figure 7.9: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ according to μ for $k \in \{k_{opt}, \hat{k}_{opt}, k_{max}, k_{the}\}$ with $\|\nabla(x_N)\|_2^2$ as the performance criterion.

Before analyzing the results, let us specify that when k increases, the number of restarts applied up to $N_{max} = 200$ (or $N_{max} = 150$ after the update of the server) iterations decreases. For example, when $k_{the} = 122$, only one restart is applied and thus, the restart does not have the opportunity to improve a lot the performance. Therefore, the results should be considered with less credit for high values of k and thus low values of μ .

First, for all performance criteria, the fast gradient method with the optimal restarting interval k_{opt} requires fewer iterations to achieve the desired precision compared to other choices of restarting interval and this makes perfect sense since this restarting interval minimizes the criterion $N_{\mu,\epsilon}(k)$.

Afterwards, compared to the classical fast gradient method, that is by taking as restarting interval k_{max} , the optimal restarting interval k_{opt} improves significantly the worst-case performance for all values of μ considered: the relative gain $G_{\mu,\epsilon}(k_{opt}, k_{max})$ is 36% on average for the performance criterion $f(x_N) - f(x_*)$. For the performance criteria $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$, the relative gain varies more according to the values of μ and it increases respectively from 40% to 65% and from 38% to 65% as μ decreases. Indeed, on Figures 7.8 and 7.9, the relative difference between $N_{\mu,\epsilon}(k_{opt})$, i.e., the red line, and $N_{\mu,\epsilon}(k_{max})$, i.e., the blue line, increases as μ decreases. Therefore, FGM-FR with the restarting interval k_{opt} improves significantly the classical fast gradient method for strongly convex functions.

Moreover, taking $k = k_{the}$ also allows to improve the performance of FGM. Indeed, the relative gain $G_{\mu,\epsilon}(k_{the}, k_{max})$ increases from 16% to 30% for $f(x_N) - f(x_*)$ and from 20% to 44% for $\|x_N - x_*\|_2^2$ as μ decreases. Thus, even by not taking the optimal value for the

restarting interval, using a restart helps to enhance the performance of the classical FGM. Note that for $\|\nabla f(x_N)\|_2^2$, the theoretical restarting interval is so bad that the number of iterations needed $N_{\mu,\epsilon}(k_{the})$ is equivalent to the one for FGM, that is $N_{\mu,\epsilon}(k_{max})$. For this criterion, the relative gain $G_{\mu,\epsilon}(k_{the}, k_{max})$ is therefore zero for all values of μ .

Finally, Figures 7.7 and 7.8 show that for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, the criteria $N_{\mu,\epsilon}(k_{opt})$ and $N_{\mu,\epsilon}(\hat{k}_{opt})$ are very close for all values of μ . This means that the optimal restarting interval formula $\hat{k}_{opt}$ in Table 7.5 approximates efficiently the optimal restarting interval k_{opt} . Therefore, using $\hat{k}_{opt}$ also allows to considerably improve the performance of the classical FGM for strongly convex functions: $G_{\mu,\epsilon}(\hat{k}_{opt}, k_{max})$ is equal to 32% on average for the performance criterion $f(x_N) - f(x_*)$ and it increases from 32% to 60% as μ decreases for the performance criterion $\|x_N - x_*\|_2^2$. Moreover, by choosing $\hat{k}_{opt}$ instead of the theoretical restarting interval k_{the} , the relative gain $G_{\mu,\epsilon}(\hat{k}_{opt}, k_{the})$ is on average respectively 10% and 20% for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. This confirms that our optimal restarting interval formula $\hat{k}_{opt}$ improves the performance of FGM more than k_{the} .

7.6 Robustness

As a reminder, we have seen in Chapters 4 and 5 that the goal of our master thesis is to determine methods that improve the performance of the fast gradient method for strongly convex functions and which are robust when the estimation of the parameter μ is not accurate. In the previous section, we have seen that FGM-FR with $\hat{k}_{opt}$ as restarting interval allows to significantly improve the performance of FGM for strongly convex functions. In this section, we will analyze and compare the robustness depending on the estimation of μ of FGM-FR using $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$ as restarting interval and FGM- μ , a version of the fast gradient method adapted to L -smooth μ -strongly convex functions but which is not robust to wrong estimations of the parameter μ . Note that we only consider the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ since we do not have derived any optimal restarting interval formula for the performance criterion $\|\nabla f(x_N)\|_2^2$.

To this end, we fix the number of iterations N and then, compare the precision reached after this number of iterations. The first step is to choose N which depends on the value of μ . Indeed, for high values of μ , the performance stagnates quite quickly due to numerical errors when the performance is less than 10^{-6} and thus, taking a too high number of iterations could lead to small differences in the performance. However, if we consider a too small number of iterations, there will not be a significant difference between the performance of FGM-FR and FGM- μ . Therefore, we consider the number of iterations $N = N_{\mu,\epsilon}(\hat{k}_{opt})$ since the performance after this number of iterations is less than 10^{-5} which is small enough while remaining greater than 10^{-6} to avoid a stagnation of the performance.

Moreover, we compute the performance of FGM-FR and FGM- μ using several estimations μ of the exact strong convexity parameter μ_* . More precisely, we consider the following

under- and overestimation of μ_* :

$$\mu \in \left\{ \frac{\mu_*}{10}, \frac{\mu_*}{5}, \frac{\mu_*}{2}, \mu_*, 2\mu_*, 5\mu_*, 10\mu_* \right\}.$$

Figures 7.10 and 7.11 show the comparison between the robustness of the two methods FGM-FR and FGM- μ .

First, we observe similar results for both performance criteria: for all values of μ_* considered, FGM-FR is more robust than FGM- μ for large underestimates, i.e., for $\mu \leq \frac{\mu_*}{5}$, and for all overestimates of μ_* , i.e., for $\mu \geq 2\mu_*$. Indeed, the performance of FGM-FR is always below the performance of FGM- μ for these values of μ . However, under- or overestimating μ_* does not have the same impact on the performance.

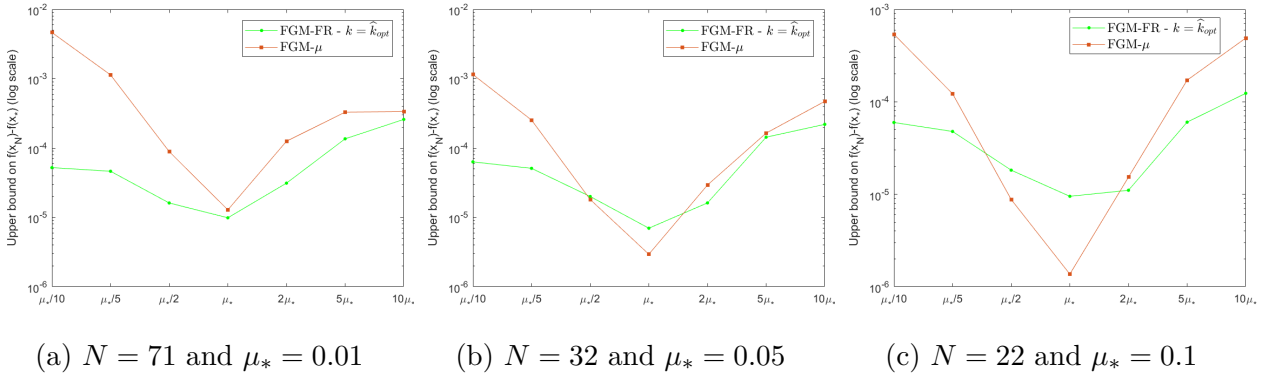


Figure 7.10: Comparison of the performance of FGM-FR with $k = \hat{k}_{opt}$ and FGM- μ after $N = N_{\mu, \epsilon}(\hat{k}_{opt})$ iterations for 1-smooth μ_* -strongly convex functions for several estimations of μ_* and $f(x_N) - f(x_*)$ as performance criterion.

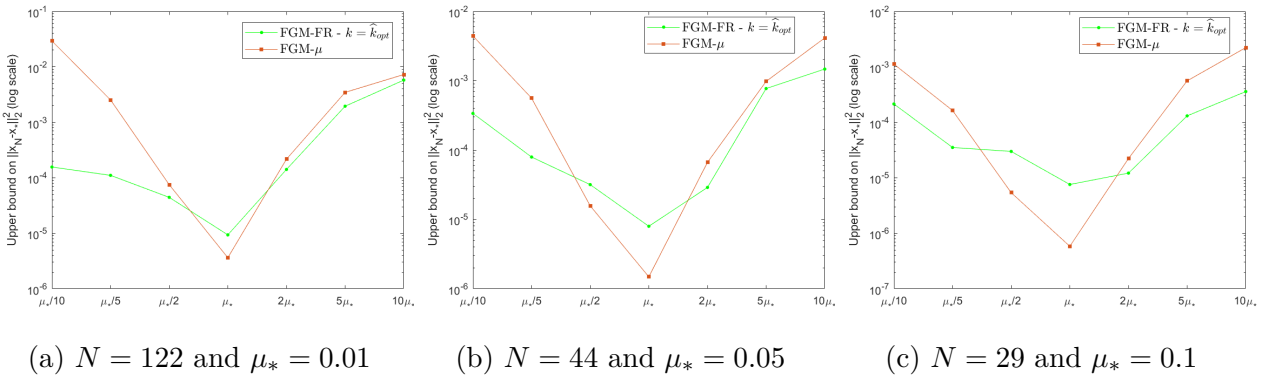


Figure 7.11: Comparison of the performance of FGM-FR with $k = \hat{k}_{opt}$ and FGM- μ after $N = N_{\mu, \epsilon}(\hat{k}_{opt})$ iterations for 1-smooth μ_* -strongly convex functions for several estimations of μ_* and $\|x_N - x_*\|_2^2$ as performance criterion.

On the one hand, overestimating the parameter μ_* , for FGM-FR leads to significant performance losses especially for small values of μ_* , i.e., $\mu_* \leq 0.05$ even if the performance

is slightly better than for FGM- μ . For example, for FGM-FR with $\mu_* = 0.01$ and the performance criterion $\|x_N - x_*\|_2^2$, taking $\mu = 10\mu_*$ gives a performance that is about 600 times worse than with the exact value of μ_* and that is almost equal to the performance obtained with FGM- $10\mu_*$ (see Figure 7.11a).

On the other hand, strongly underestimating the value μ_* seems much more robust than overestimating it, especially for small values of μ_* . For example, for FGM-FR with $\mu_* = 0.01$ and the performance criterion $f(x_N) - f(x_*)$, taking $\mu = \frac{\mu_*}{10}$ gives a performance that is only about 5 times worse than with the exact value of μ_* but is about 90 times better than the performance obtained with FGM- $\frac{\mu_*}{10}$ (see Figure 7.10a). This allows using conservative estimates when μ_* is unknown without too much impact on the performance which is quite interesting since users tend to use too conservative estimates. However, if the underestimation of μ_* is small, i.e., $\mu = \frac{\mu_*}{2}$, FGM- μ gives slightly better performance for $\mu_* \geq 0.05$. Moreover, if the function is better conditioned around the minimum, FGM-FR will take advantage of this better conditioning and work better around the minimum. Indeed, if the function is better conditioned around the minimum, it amounts to underestimate locally μ_* and since FGM-FR is robust to underestimation of μ_* , FGM-FR will work better close to the minimum.

Finally, we can see that FGM- μ performs better than FGM-FR when μ_* is known exactly, except for $\mu_* = 0.01$ and $f(x_N) - f(x_*)$ as performance criterion but in this case, the performance of both methods is close. This is logical since FGM- μ was specifically designed for strongly convex functions.

To conclude, in a general way, FGM-FR with $\hat{k}_{opt}$ allows to improve the performance of FGM- μ for wrong estimations of the exact value of the strong convexity parameter, especially for large underestimations, but FGM- μ works better when the estimation of the parameter μ_* is exact. It is, therefore, better to use FGM-FR with $\hat{k}_{opt}$ when μ_* is not known, especially when a conservative estimate of μ is used, and it is better to use FGM- μ when μ_* is known exactly.

7.7 Conclusion

To conclude, we have first observed in this chapter that the fast gradient method with fixed restart helps to improve the worst-case performance of the classical fast gradient method for strongly convex functions when the restarting interval is well chosen. Afterwards, we have evaluated the optimal restarting interval k_{opt} , i.e., the value of the restarting interval k which gives the best worst-case performance, for a large range of values of μ based on a specific criterion $N_{\mu,\epsilon}(k)$ for the three performance criteria considered. Then, we have derived a formula $\hat{k}_{opt}$ depending on L and μ that allows estimating k_{opt} for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$: $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$. However, we did not find a formula for the performance criterion $\|\nabla f(x_N)\|_2^2$.

Afterwards, the relative gain has been analyzed: compared to the classical FGM, FGM-FR with $k = \hat{k}_{opt}$ reduces the number of iterations needed to reach a specific precision by on average 32% for the performance criterion $f(x_N) - f(x_*)$ and from 32% to 60% as μ

decreases for the performance criterion $\|x_N - x_*\|_2^2$, which is quite a significant improvement.

Finally, the robustness of FGM-FR with $k = \hat{k}_{opt}$ has been analyzed for under- or over-estimates of the exact strong convexity parameter μ_* and compared with the robustness of FGM- μ . We have observed that in addition to improving performance, FGM-FR with $k = \hat{k}_{opt}$ is also robust when the estimation of the strong convexity parameter μ_* is not accurate, especially for large underestimations of μ_* . Unfortunately, if the μ_* is overestimated, there will be some performance losses for FGM-FR but not as many as for FGM- μ . It is nevertheless better to use FGM- μ when μ_* is known exactly.

Therefore, we have found a version of the fast gradient method that takes advantage of the strong convexity for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ and that is robust even for wrong estimations of the strong convexity parameter.

8

Fast gradient method with adaptive restart

This chapter first explains the fast gradient method with adaptive restart, then details how this method can be implemented using the toolbox PESTO and analyzes the worst-case performance of this method. Finally, considering a particular case of the fast gradient method with adaptive restart, we show that this method does not improve the performance of the classical fast gradient method.

8.1 Method

As explained in Chapter 5, Candès and O’Donoghue proposed another strategy of restart: the adaptive restart. This strategy consists of applying a restart when the function does not evolve sufficiently according to a certain criterion: for example, when the function value starts to increase or when the momentum is in the « wrong » direction. In practice, the adaptive restart sets the parameter $\theta_{i+1} = 1$ when a specific restarting condition is met. The algorithm of the fast gradient method with adaptive restart, i.e., FGM-AR, can be described as follows.

FGM-AR

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$ and $\theta_0 = 1$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$$

$$\theta_{i+1} = \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$$

$$x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}}(y_{i+1} - y_i)$$

If a restarting condition holds

$$\theta_{i+1} = 1$$

In this master thesis, we have implemented the two following restarting conditions:

- **Function scheme:** restart whenever

$$f(y_{i+1}) > f(y_i). \quad (\text{FS})$$

- **Gradient scheme:** restart whenever

$$\langle \nabla f(x_i), y_{i+1} - y_i \rangle > 0. \quad (\text{GS})$$

As mentioned in Chapter 5, these schemes were both introduced by Candès and O’Donoghue in [OC15].

These two restarting schemes are linked by the following proposition.

Proposition 8.1. *Suppose the convex function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ and the iterates x_i , y_i and y_{i+1} of FGM-AR. If $x_i = y_i$, then we have*

$$\text{If } (\text{GS}) \text{ holds} \Rightarrow (\text{FS}) \text{ holds.}$$

Proof. If $x_i = y_i$, (GS) can be written as $\langle \nabla f(x_i), y_{i+1} - y_i \rangle = \langle \nabla f(y_i), y_{i+1} - y_i \rangle$ and thus, if (GS) is satisfied, we know that:

$$\langle \nabla f(y_i), y_{i+1} - y_i \rangle > 0.$$

Then, by using Definition 2.2 of a convex function, we can find that:

$$\begin{aligned} f(y_{i+1}) &\geq f(y_i) + \nabla f(y_i)^T (y_{i+1} - y_i) \\ \Leftrightarrow f(y_{i+1}) - f(y_i) &\geq \langle \nabla f(y_i), y_{i+1} - y_i \rangle > 0 \end{aligned}$$

Therefore, (FS) is satisfied. $\square$

Proposition 8.1 demonstrates that when $x_i = y_i$, if the gradient scheme (GS) condition holds, then the function scheme (FS) condition also holds. On the contrary, when $x_i = y_i$, if (FS) is not satisfied, then (GS) will not be satisfied either.

8.2 Implementation

The FGM-AR algorithm cannot directly be implemented in PESTO because of the restarting conditions (FS) and (GS). Indeed, as Section 3.4 mentioned, PESTO cannot handle the strict inequalities nor the If conditions on variables. Therefore, we need to find a way to overcome these issues.

8.2.1 If condition

Since the If condition on variables cannot be applied with PESTO, we need to find a trick to remove this condition. A way is to compute the result for all possible outcomes of this condition and take the maximum result since we are looking for a worst-case performance, i.e., a maximum performance. Thus, to obtain the worst-case performance of FGM-AR, we compute the worst-case performance for all the possible combinations of restart/no

restart and take the maximum of the worst-case performance of all these cases. More precisely, we compute the performance of all possible cases where a restart is triggered or not at each iteration, i.e., where the restarting condition (FS) or (GS) is satisfied or not at each iteration. In practice, to achieve this, we constrain the restarting condition (FS) or (GS) to be respected or not and according to those constraints, a restart is applied or not, i.e., θ is set to 1 or not.

Moreover, we do not consider an adaptive restart at the last iteration since this restart would not improve the worst-case performance. Indeed, if a restart is applied at the last iteration, θ_N is set to 1 but does not impact the iterates.

To better understand how the performance of FGM-AR can be obtained using PESTO, let us consider the following example: we want to compute the worst-case performance of FGM-AR with the (FS) condition and after 4 iterations. Therefore, from iteration 1 to iteration 3, we need to consider the case where the restart is applied, i.e., the (FS) condition is satisfied and θ is set to 1, or the case where the restart is not applied, i.e., the (FS) condition is not satisfied. We refer to these cases respectively as 1 and 0: a 1 (respectively a 0) in the n^{th} position means that the restart at iteration n is applied (respectively is not applied). Then, Figure 8.1 illustrates how the problem is subdivided: first, we need to consider a restart or not at the first iteration, then at the second iteration until the $(N - 1)^{th}$ iteration and the last line gives all the cases that must be computed to determine the worst-case performance of FGM-AR. In our example, we thus have to compute the worst-case performance of 8 cases.

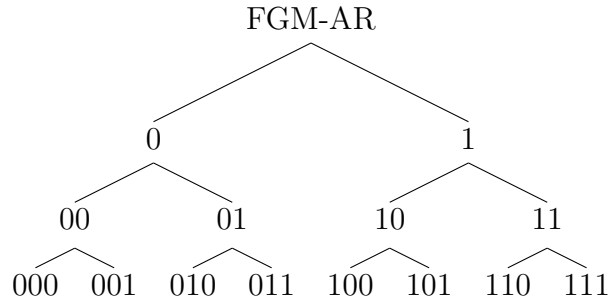


Figure 8.1: Decomposition in cases to compute the worst-case performance of FGM-AR after 4 iterations. The "0" (resp. "1") in n^{th} position means the n^{th} restart, i.e., the restart at iteration n , is not applied (resp. is applied).

More precisely, we have to compute the performance of FGM-AR by adding the following constraints for each case, with $1 \leq i + 1 \leq 3$:

- 1 in $(i + 1)^{th}$ position $\Leftrightarrow$ apply a restart at iteration $i + 1$: adding the constraint $f(y_{i+1}) > f(y_i)$ and set θ_{i+1} to 1.
- 0 in $(i + 1)^{th}$ position $\Leftrightarrow$ do not apply the restart at iteration $i + 1$: adding the constraint $f(y_{i+1}) \leq f(y_i)$.

Finally, the performance of FGM-AR is given by the worst of the worst-case performance of all these cases, i.e., the maximum performance among these 8 cases.

To prove that this reasoning is indeed correct and provides the worst-case performance of the fast gradient method with adaptive restart, let us write the following proposition:

Proposition 8.2. *Suppose the problem*

$$z = \max_{x \in S \cap T} \mathcal{P}(x).$$

Let $S = S_1 \cup S_2 \cup \dots \cup S_n$ be a partition of S . Then

$$z = \max_{i \in \{1, \dots, n\}} \max_{x \in S_i \cap T} \mathcal{P}(x).$$

Proof. Since $S = S_1 \cup S_2 \cup \dots \cup S_n$, at least one of the sub-domains contains the maximum, i.e., $\exists j \in \{1, \dots, n\}$ such that

$$\begin{aligned} \max_{x \in S_j \cap T} \mathcal{P}(x) &\geq \max_{x \in S_i \cap T} \mathcal{P}(x) \quad \forall i \in \{1, \dots, n\} \\ \Leftrightarrow \max_{x \in S_j \cap T} \mathcal{P}(x) &= \max_{i \in \{1, \dots, n\}} \max_{x \in S_i \cap T} \mathcal{P}(x) = z. \end{aligned}$$

□

This proposition can be applied to our problem. Indeed, our problem is divided into several cases by imposing a restart or not. Thus, the domain of each case is a sub-domain of the initial problem and all the sub-domains correspond to the entire domain since we consider all possible combinations of restart/no restart. Then, the worst-case performance, i.e., the maximum of our problem, is given by the maximum of all the obtained worst-case performance from each case.

Afterwards, the number of cases to compute grows exponentially with the number of iterations N : when N increases, the number of cases to consider increases since at each iteration, we must consider the cases where we apply a restart or not. Thus, we must solve 2^{N-1} cases with different constraints to obtain the worst-case performance of the fast gradient method with adaptive restart after N iterations. The -1 comes from the fact that no restart is applied at the last iteration. To be able to consider a larger number of iterations N , we consider a restarting interval k : this allows us to add a constraint of restart or no restart only every k iterations. The number of cases to solve then becomes $2^{\lceil \frac{N}{k} \rceil - 1}$.

However, not all possible combinations of restart/no restart are feasible depending on the restarting condition used. To show that, let us simplify and analyze the iterations of the fast gradient method at the beginning and after a restart. The three first iterations are presented below and the iterations after a restart are already explained in Section [7.1](#). As a reminder, an usual gradient step means that the gradient step from the same variable, i.e., $y_{i+1} = y_i - \frac{1}{L} \nabla f(y_i)$ or $x_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$.

iteration 0	$y_0 = x_0$	
	$\theta_0 = 1$	
iteration 1	$y_1 = y_0 - \frac{1}{L}\nabla f(y_0)$	Usual gradient step
	$x_1 = y_1$	
iteration 2	$y_2 = y_1 - \frac{1}{L}\nabla f(y_1)$	Usual gradient step
	$x_2 = y_2 + \frac{\theta_1-1}{\theta_2}(y_2 - y_1)$	
iteration 3	$y_3 = x_2 - \frac{1}{L}\nabla f(x_2)$	
	$x_3 = y_3 + \frac{\theta_2-1}{\theta_3}(y_3 - y_2)$	
	$\vdots$	

We can observe that at the two first iterations and two iterations after a restart (see Section 7.1), the variable y is given by an usual gradient step. Therefore, we know that the function value must decrease at these steps due to the Descent Lemma 2.12:

$$f(y_{i+1}) - f(y_i) \leq -\frac{1}{2L}\|\nabla f(y_i)\|_2^2 \leq 0.$$

This inequality means that the (FS) criterion is infeasible for these iterations and therefore, the (GS) criterion is not feasible either, as demonstrated by Proposition 8.1. Thus, we can conclude that at least the following cases are infeasible for both the function scheme (FS) and the gradient scheme (GS):

1. the cases where a restart is imposed at the first iteration.
2. the cases where a restart is imposed at the second iteration.
3. the cases where a restart is imposed two iterations after another restart.

Therefore, only the feasible cases can be considered in the computation of the worst-case performance of the fast gradient method with adaptive method. This allows reducing the number of computations needed to compute the worst-case performance of FGM-AR. Indeed, for example, simply by considering no restart at the two first iterations, computing the worst-case performance of FGM-AR with the restarting interval $k = 1$ and N iterations requires 2^{N-3} computations instead of 2^{N-1} .

8.2.2 Strict inequality

As previously mentioned, PESTO cannot handle the strict inequalities and therefore, these inequalities must be transformed into non-strict ones. To this end, we add a $\delta \geq 0$ term which allows to remove the strict constraint: $f(y_{i+1}) > f(y_i)$ becomes $f(y_{i+1}) \geq f(y_i) + \delta$ and $\langle \nabla f(x_i), y_{i+1} - y_i \rangle > 0$ becomes $\langle \nabla f(x_i), y_{i+1} - y_i \rangle \geq \delta$.

Note that the case $\delta = 0$ is actually not admissible since $f(y_{i+1}) = f(y_i)$ or $\langle \nabla f(x_i), y_{i+1} - y_i \rangle = 0$ should not trigger a restart but this case is interesting to analyze since it gives

an upper bound on the worst-case performance. Indeed, as δ decreases, the domain over which we maximize becomes larger and thus, the smallest value of δ gives an upper bound.

The choice of δ could have an impact on the worst-case performance: this will be analyzed in the next section.

8.3 Worst-case performance analysis

In this section, we analyze the worst-case performance of FGM-AR with the (FS) and (GS) restarting conditions for several values of δ and μ . Then, we compare it with the worst-case performance of the classical FGM. To this end, we have chosen three values of δ and two values of μ in order to determine the influence of these parameters:

$$\delta \in \{10^{-3}, 10^{-5}, 0\} \text{ and } \mu \in \{0.01, 0.05\}.$$

Then, we restrict ourselves to $N = 30$ iterations and a restarting interval $k = 5$. These values allow having a good compromise between speed while having a sufficiently high number of iterations. Indeed, to obtain the worst-case performance at the iterations 26 to 30, we have to compute $2^{\frac{30}{5}-1} = 2^5 = 32$ cases, which is rather consequential in term of computational time.

Figures 8.2 and 8.3 show our results. Note that since the results for the performance criterion $\|\nabla f(x_N)\|_2^2$ are quite similar to the ones for $f(x_N) - f(x_*)$, only the results for $f(x_N) - f(x_*)$ are displayed in this section but the results for $\|\nabla f(x_N)\|_2^2$ are available in Appendix 8.A.

In a general way, for all values of μ and δ , Figures 8.2 and 8.3 show that FGM-AR does not seem to improve the performance of the classical FGM. Indeed, for $\mu = 0.01$, the performance of FGM-AR seems to be worse than FGM, especially for the performance criterion $\|x_N - x_*\|_2^2$ (see Figure 8.3a). For $\mu = 0.05$, FGM-AR gives results close to FGM in terms of worst-case performance for all values of δ . Therefore, it seems that the lower the μ , the worse the performance of FGM-AR is compared to the classical FGM.

Afterwards, we observe that the value of δ influences the worst-case performance, especially for the performance criterion $\|x_N - x_*\|_2^2$ (see Figure 8.3). Indeed, it seems that the bigger the value of δ , the better the worst-case performance is. By analyzing the results provided by PESTO, it seems that a lot of numerical or infeasible problems appear as δ increases when we solved the different cases. This seems logical since when δ increases, the (FS) or the (GS) condition becomes more restrictive. Thus, the solver may have more difficulty in solving the problem or come across an infeasible case. However, the case where we imposed that no adaptive restart is triggered, is always successfully solved for all values of δ considered.

Note that it is important not to confuse numerical errors and numerical problems. On the one hand, numerical errors appear after a certain number of iterations when the worst-case performance is close to 10^{-6} : the problem is successfully solved by the solver but the precision of the solver is reached. On the other hand, numerical problems appear because of a break in the computation of the solution: the problem is not successfully solved but the solver still returns a wrong solution.

For example, for the performance criterion $f(x_N) - f(x_*)$, numerical problems are noticeable on Figure 8.2 for $\delta = 10^{-3}$ where an unexpected change appear after iteration 20 and then, the performance stabilize around some value. Concerning the performance criterion $\|x_N - x_*\|_2^2$, numerical problems are not apparent since, for this performance criterion, PESTO returns negative values when numerical problems appear and, thus, these values are not considered as the worst-case performance since they are not the biggest one among all restarting cases of FGM-AR.

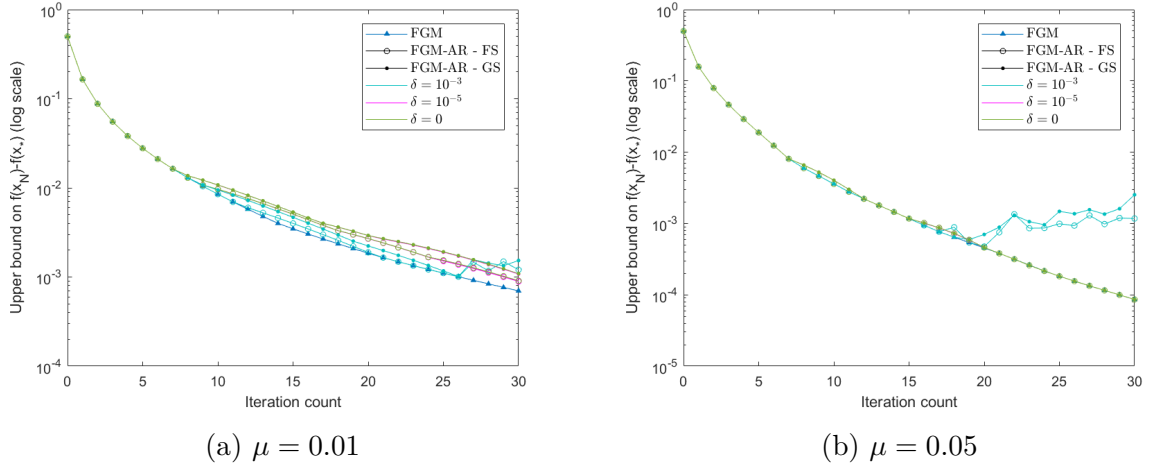


Figure 8.2: Comparison between the worst-case performance of FGM and FGM-AR with the (FS) and (GS) condition, for several values of δ , for 1-smooth μ -strongly convex functions and $f_N - f_*$ as performance criterion.

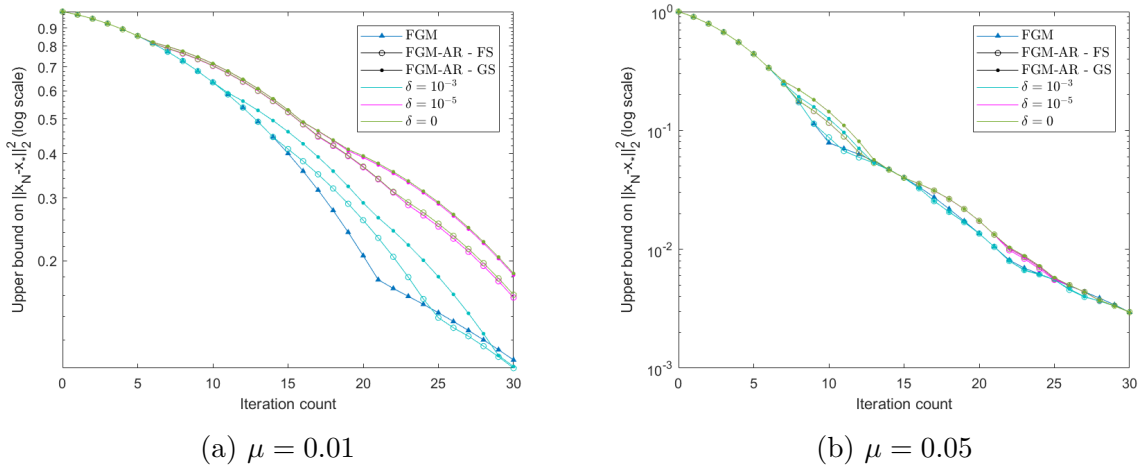


Figure 8.3: Comparison between the worst-case performance of FGM and FGM-AR with the (FS) and (GS) condition, for several values of δ , for 1-smooth μ -strongly convex functions and $\|x_N - x_*\|_2^2$ as performance criterion.

In fact, by changing the strict inequality into a non-strict one using a δ term, the domain becomes closed and by analyzing the performance when δ varies, it seems that the lower

the value of δ , the worse the performance. This would mean that the solutions seem to tend to the limit of the domain, i.e., the solution satisfies $f(y_{i+1}) = f(y_i) + \delta$ for the condition (FS) or $\langle \nabla f(x_i), y_{i+1} - y_i \rangle = \delta$ for the condition (GS). Therefore, the problem of finding the performance of FGM-AR is ill-defined (i.e., has no solution) since for both conditions the solution approaches the boundary of the domain as δ decreases to zero.

Therefore, solving FGM-AR with $\delta = 0$ could give a supremum in the worst-case performance. Indeed, setting $\delta = 0$ gives an upper bound on the worst-case performance of FGM-AR as explained in Subsection 8.2.2. Moreover, this solution is not acceptable when the solution is given by the boundary of the domain. Indeed, for $\delta = 0$, we have $f(y_{i+1}) = f(y_i)$ or $\langle \nabla f(x_i), y_{i+1} - y_i \rangle = 0$ when a restart is imposed but this case should actually not trigger a restart. Therefore, when the condition (FS) or (GS) gives an equality with $\delta = 0$, this is not a feasible solution for FGM-AR but only an upper bound.

Finally, we observe that the worst-case performance for FGM-AR with the (FS) scheme is slightly better, i.e., less than or equal to the one with the (GS) scheme for all the values of μ and δ , except when numerical problems occur.

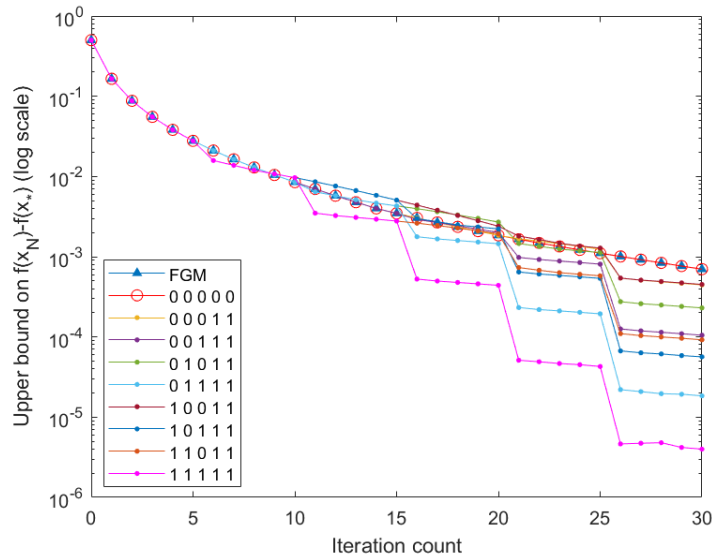


Figure 8.4: Comparison of the worst-case performance of FGM and several cases of FGM-AR with the (FS) condition, for $\delta = 0$, $\mu = 0.01$ and $f_N - f_*$ as performance criterion. The "0" (resp. "1") in n^{th} position means that the n^{th} restart is not applied (resp. is applied).

In order to understand how the different cases of restart/no restart behave, we analyze the performance evolution of some of the adaptive cases of FGM-AR with the (FS) condition, $\delta = 0$, $\mu = 0.01$ and the performance criterion $f(x_N) - f(x_*)$ on Figure 8.4. We see that all the cases whose performance is worse than FGM contain at least one restart. This can be explained by the fact that the case where we imposed that no restart is triggered, i.e., the case $00 \dots 0$, gives a feasible solution for FGM since θ is never set to 1 in this case. Therefore, FGM is at least equal to the worst-case performance of the case where no restart is triggered.

Moreover, we observe that the performance of the case $00\cdots 0$ in which no restart is applied seems equal to the performance of the classical FGM. If the performance of a particular case of FGM-AR, i.e., the case without restart, is almost the same as the performance of the classical FGM, this would show that FGM-AR cannot improve much the performance of the classical FGM. Therefore, an approach is to compare the performance of the case without restart and the classical FGM: this will be explored in the next section.

8.4 Inefficiency of the adaptive restart

In the previous sections, we have seen that the fast gradient method with adaptive restart FGM-AR is more complicated to implement than expected. Indeed, on the one hand, since we cannot have an If condition on variables with PESTO, we need to separate the computation of the performance of the adaptive restart into several cases: more precisely, we need to compute the worst-case performance for each possible combination of restart/no restart. Thus, the number of computations grows exponentially as the number of iterations N increases or as the restarting interval k decreases. On the other hand, using a δ term to remove the strict inequality gives a not well-defined problem.

Then, we have also noticed that the performance of the case in which no adaptive restart is triggered seems to be always close to the performance of FGM. We will refer to this case as "FGM-AR without restart". Note that this particular case gives a lower bound on the performance of FGM-AR since the worst-case performance of FGM-AR is given by the maximum of all the worst-case performance of the different cases: one particular case, therefore, gives a lower bound on the performance of FGM-AR. The idea is thus to compare the performance between the classical FGM and FGM-AR without restart since this method allows to avoid the two following problems encountered to implement FGM-AR:

- **If condition:** by considering only one case, i.e., the case FGM-AR without restart, we do not need to compute the worst-case performance in all possible cases. Therefore, we can compute the performance of FGM-AR without restart for a much higher number of iterations N and with an adaptive restart at every iteration, i.e., $k = 1$ (in our case, no restart at every iteration).
- **Strict inequality:** the case with no restart does not have a strict inequality and, therefore, we do not need to use a δ term.

We apply this idea for the same values of μ as above and for all performance criteria considered and compare the performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-AR without restart. Then, to keep a reasonable computational time, we limit ourselves to $N = 80$ iterations. Since the results for the performance criteria $f(x_N) - f(x_*)$ and $\|\nabla f(x_N)\|_2^2$ are quite similar, the results for $f(x_N) - f(x_*)$ are shown in this section but the results of $\|\nabla f(x_N)\|_2^2$ are available in Appendix 8.B. Figures 8.5 and 8.6 show the results.

First, we can see that the worst-case performance of FGM-AR without restart seems almost equal for both conditions (FS) and (GS) since the worst-case performance curves appear to be superimposed on all the figures except when there are numerical errors.

Indeed, as already mentioned in the previous chapters, when the worst-case performance is close to 10^{-6} , numerical errors appear and the worst-case performance starts to oscillate around a certain value.

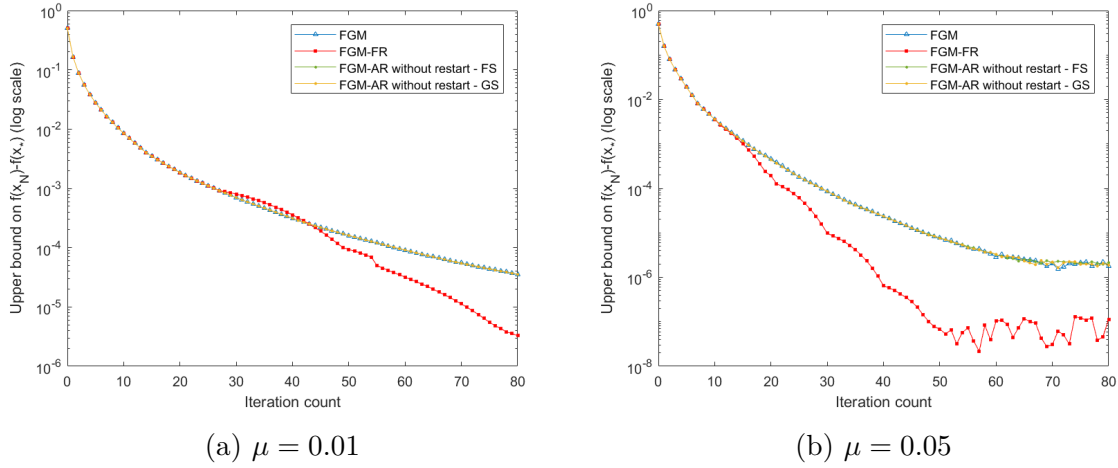


Figure 8.5: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-AR without restart with the (FS) and (GS) conditions for 1-smooth μ -strongly convex functions and $f_N - f_*$ as performance criterion.

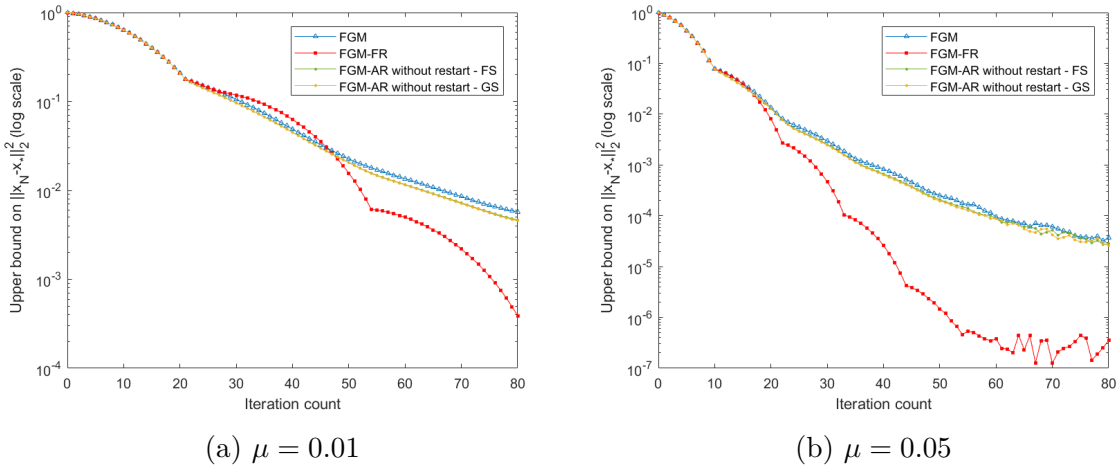


Figure 8.6: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-AR without restart with the (FS) and (GS) conditions for 1-smooth μ -strongly convex functions and $\|x_N - x_*\|_2^2$ as performance criterion.

Then, in a general way, FGM-AR without restart performs very similarly than the classical FGM. Indeed, we observe that the worst-case performance of FGM-AR without restart and FGM is close for both conditions (FS) and (GS), for all values of μ considered and for all performance criteria, especially for $f(x_N) - f(x_*)$ and $\|\nabla f(x_N)\|_2^2$. For these two performance criteria, no difference can be distinguished between the performance of FGM and FGM-AR without restart. However, for the performance criterion $\|x_N - x_*\|_2^2$, there

is a small visible difference between the worst-case performance of FGM and FGM-AR but FGM-AR is significantly worse than FGM-FR.

Therefore, these results show numerically that the worst-case performance of the FGM-AR is inefficient in terms of worst-case performance since a lower bound on the performance of FGM-AR is already really close to the performance of FGM.

Afterwards, we would like to analyze how evolves the difference between the worst-case performance of FGM and FGM-AR without restart. To this end, we compute the relative difference between the performance of FGM and FGM-AR without restart for both conditions (FS) and (GS). Figures 8.7 and 8.8 show the results respectively for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ (see Appendix 8.B for the results with $\|\nabla f(x_N)\|_2^2$ as performance criterion).

First, on both figures, we can observe that there is no difference between FGM and FGM-AR without restart for both schemes until a certain number of iterations. Indeed, in this case, the worst-case function obtained by solving FGM does not trigger the (FS) or (GS) condition. Therefore, imposing that no adaptive restart is applied, i.e., imposing that (FS) or (GS) is not satisfied, gives the same worst-case performance as FGM since the condition (FS) or (GS) does already not hold for the worst-case function of FGM.

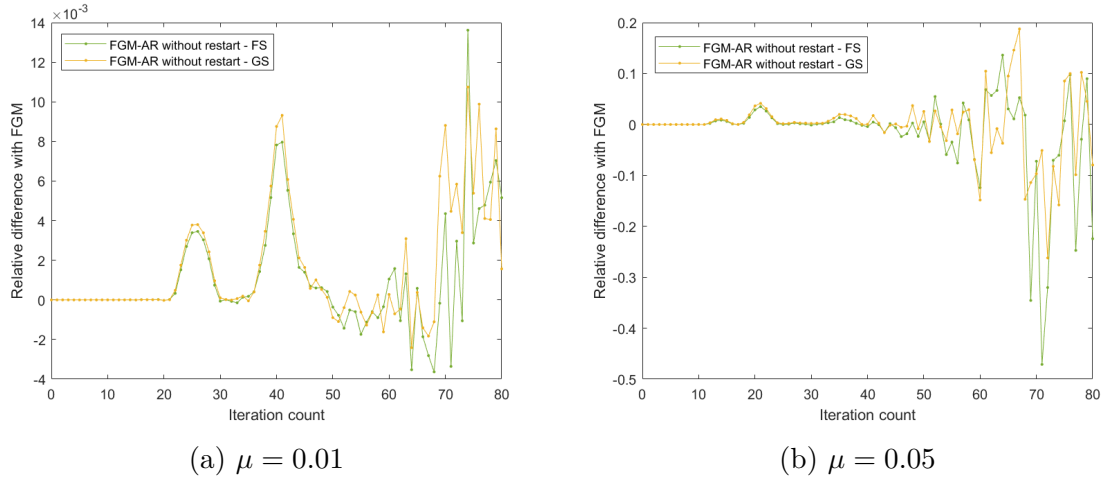


Figure 8.7: Relative difference between the worst-case performance of FGM and FGM-AR without restart with the (FS) and (GS) conditions for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

Then, after a certain number of iterations, a difference appears between the performance of FGM and FGM-AR without restart. This corresponds to the case where the restarting conditions are triggered in the worst-case function obtained with FGM: their worst-case performance is thus no longer equal. However, the relative difference evolves in various ways depending on the performance criterion.

On the one hand, for the performance criterion $f(x_N) - f(x_*)$ (and similarly for the performance criterion $\|\nabla f(x_N)\|_2^2$), Figure 8.7 shows that the relative difference between the two methods is quite small, i.e., less than 0.1, for both values of μ and for both

restarting schemes without considering numerical errors. On the other hand, the relative difference increases slowly for the performance criterion $\|x_N - x_*\|_2^2$ but stays relatively small, i.e., less than 0.35, for both values of μ and both restarting schemes.

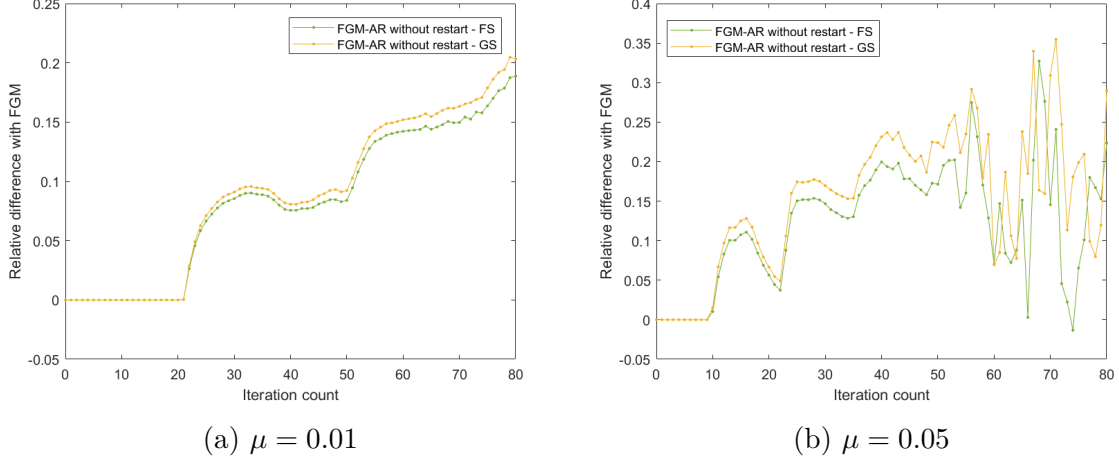


Figure 8.8: Relative difference between the worst-case performance of FGM and FGM-AR without restart with the (FS) and (GS) conditions for 1-smooth μ -strongly convex functions and $\|x_N - x_*\|_2^2$ as performance criterion.

Therefore, this confirms that FGM-AR does not allow to improve the performance of FGM, even when the schemes are triggered in the worst-case function of FGM after a certain number of iterations.

Finally, we can see that the relative difference is always bigger for the condition (GS) than for the condition (FS), except when numerical errors occur. This means that the worst-case performance is slightly better, i.e., lower, for FGM-AR without restart using the condition (GS) than using the condition (FS).

8.5 Conclusion

To conclude, we have seen in this chapter that the fast gradient method with adaptive restart is quite complicated to implement with the toolbox PESTO. On the one hand, since PESTO cannot deal with If conditions on variables, the computation of the worst-case performance of FGM-AR must be separated into several cases corresponding to every possible combination of restart/no restart. The worst-case performance of FGM-AR is then given by the worst performance among the worst-case performance of all cases. Thus, the number of cases grows exponentially as the number of iterations increases or as the restarting interval decreases. On the other hand, to remove the strict inequality, we use a δ term but this term makes the problem of finding the worst-case performance of FGM-AR ill-defined: indeed, the δ parameter has a significant influence on the worst-case performance of FGM-AR.

Then, we have observed that the worst-case performance of FGM-AR without restart,

i.e., the case where no adaptive restart is triggered, and the classical FGM seemed to be very close. Thus, we can get rid of the implementation issues by comparing this particular case, which gives a lower bound on the worst-case performance of FGM-AR, and FGM. Indeed, implementing and computing the worst-case performance of FGM-AR without restart is much simpler: we do not need to use a δ term and we can compute the worst-case performance for a large number of iterations with the restarting interval $k = 1$ since we focus only on one particular case. Therefore, since the performance of FGM-AR without restart and the classical FGM are close for both restarting conditions and all performance criteria considered, we can conclude that FGM-AR does not allow to improve the performance of FGM and is inefficient in terms of the worst-case.

8.A Worst-case performance analysis for the performance criterion $\|\nabla f(x_N)\|_2^2$

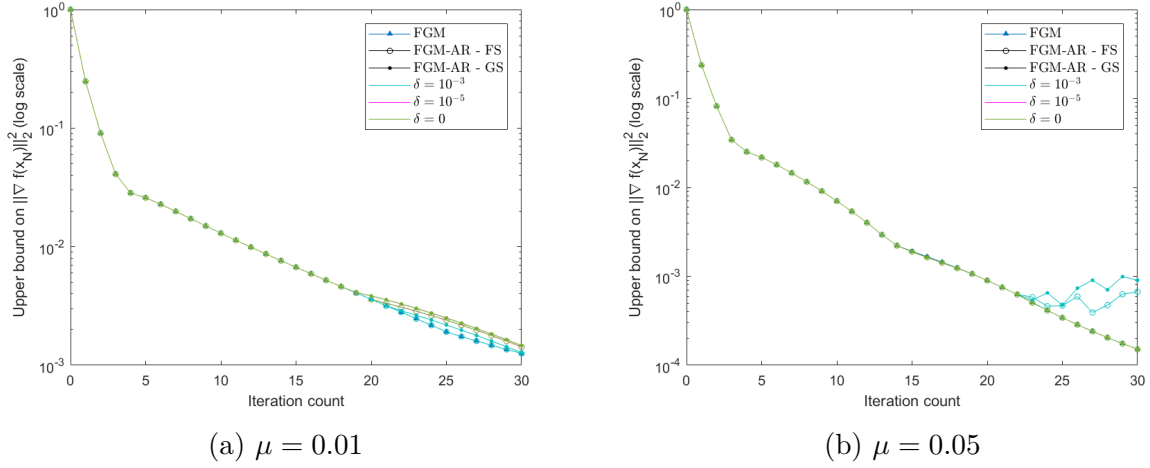


Figure 8.9: Comparison between the worst-case performance of FGM and FGM-AR with the (FS) and (GS) conditions, for several values of δ , for 1-smooth μ -strongly convex functions and $\|\nabla f(x_N)\|_2^2$ as performance criterion.

8.B Inefficiency of the adaptive restart for the performance criterion $\|\nabla f(x_N)\|_2^2$

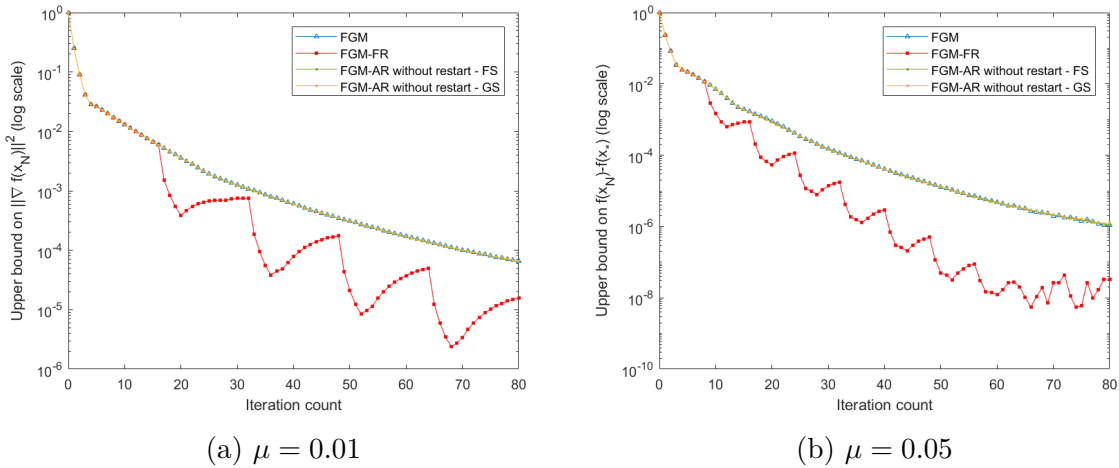
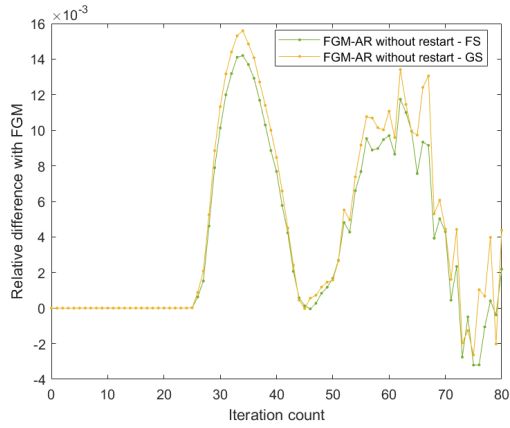
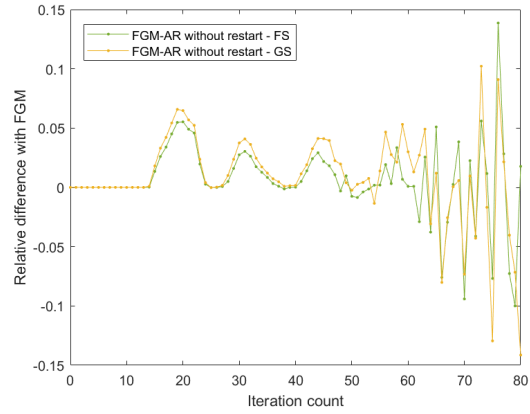


Figure 8.10: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-AR without restart with the (FS) and (GS) conditions, for 1-smooth μ -strongly convex functions and $\|\nabla f(x_N)\|_2^2$ as performance criterion.



(a) $\mu = 0.01$



(b) $\mu = 0.05$

Figure 8.11: Relative difference between the worst-case performance of FGM and FGM-AR without restart with the (FS) and (GS) conditions, for 1-smooth μ -strongly convex functions and $\|\nabla f(x_N)\|_2^2$ as performance criterion.

9

Fast gradient method with soft restart

In Chapter 7, we have seen that the fixed restart allows to considerably improve the worst-case performance of the classical fast gradient method. The goal of this chapter is to determine a new efficient approach for restarting the fast gradient method by using a soft restart. Several strategies of soft restart have been implemented and analyzed to determine whether these strategies improve the performance of the classical fast gradient method and the fast gradient method with fixed restart. Finally, the promising strategy(ies) have been explored in more detail.

9.1 Method

The general idea of the soft restart is to apply a fraction of restart at each iteration instead of applying a «full» restart after a constant number of iterations as the fixed restart does. Thus, we define a parameter

$$\alpha \in [0, 1] \text{ such that } \alpha = \begin{cases} 0 & \text{if no restart is applied,} \\ 1 & \text{if a «full» restart is applied,} \end{cases} \quad (9.1)$$

where the parameter α is called the restarting fraction in this chapter.

In practice, in order to perform a soft restart on the fast gradient method, we can vary the formula of the momentum parameter θ . More precisely, we use the parameter α to control the evolution of the momentum parameter θ as follows:

$$\theta_{i+1} = \alpha \cdot 1 + (1 - \alpha) \cdot \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}.$$

Therefore, the conditions of the restarting fraction (9.1) are respected since we have

$$\begin{aligned} \alpha = 0 &\Leftrightarrow \theta_{i+1} = \frac{1 + \sqrt{4\theta_i^2 + 1}}{2} \Leftrightarrow \text{no restart is applied,} \\ \alpha = 1 &\Leftrightarrow \theta_{i+1} = 1 \Leftrightarrow \text{a «full» restart is applied.} \end{aligned}$$

Then, three strategies with different values for the restarting fraction α have been developed and explained in the following subsections.

9.1.1 First strategy

The idea of this first strategy is to control the increase in θ by fixing the restarting fraction to $\alpha = \frac{1}{k}$ which allows applying a k^{th} of restart at each iteration. Additionally, the momentum parameter θ is reset to 1 every k iterations as for the fixed restart.

This first strategy is fully described in the following algorithm.

FGM-SR1

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$, $\theta_0 = 1$ and restarting interval $k \geq 1$

$\alpha = \frac{1}{k}$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$$

$$\theta_{i+1} = \alpha + (1 - \alpha) \cdot \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$$

$$x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}} (y_{i+1} - y_i)$$

If $(i + 1) \bmod k == 0$

$$\theta_{i+1} = 1$$

9.1.2 Second strategy

For the second strategy, the idea is to apply a restart of increasing magnitude by increasing α and then, resetting α to 0 every k iterations. More concretely, we want the restarting fraction α to increase from 0 to 1 for k iterations and then, α is reset to 0 and starts to increase again. Thus, the closer an iteration is to a multiple of the restarting interval k , the larger a partial restart is applied. Therefore, we want to determine the restarting fraction α that follows these conditions:

$$\begin{cases} \alpha_{(j-1)k+1} = 0 & \text{(no restart)} \\ \alpha_{jk} = 1 & \text{(full restart)} \\ \text{increasing } \alpha_i \text{ for } i \in \{(j-1)k + 1, \dots, jk\} \end{cases} \quad \forall j \in \{1, 2, \dots, \left\lceil \frac{N}{k} \right\rceil\}$$

To achieve this, two formula have been implemented using the operator modulo. The first one increases gradually while the second one increases exponentially.

$$\alpha_{i+1} = \frac{i \bmod k}{k - 1} \tag{a}$$

$$\alpha_{i+1} = \frac{e^{\frac{i \bmod k}{100}} - 1}{e^{\frac{k-1}{100}} - 1} \tag{b}$$

The strategy is implemented in the following algorithm.

FGM-SR2a/b

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$, $\theta_0 = 1$ and restarting interval $k \geq 1$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$$

$$\alpha_{i+1} = \text{formula (a) or (b)}$$

$$\theta_{i+1} = \alpha_{i+1} + (1 - \alpha_{i+1}) \cdot \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$$

$$x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}}(y_{i+1} - y_i)$$

9.1.3 Third strategy

This third strategy is very similar to the first one but it does not apply a «full» restart, i.e., the momentum parameter θ is not set to 1, every k iterations. Instead, a restarting fraction is just set to $\alpha = \frac{1}{p}$. This allows analyzing whether a soft restarting scheme without imposing a «full» restart every k iterations gives better worst-case performance than FGM and FGM-FR. Note that the larger the value of the restarting factor p , the lower the soft restart applied at each iteration.

This strategy is implemented in the following algorithm.

FGM-SR3

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, $y_0 = x_0 \in \mathbb{R}^d$, $\theta_0 = 1$ and restarting factor $p \geq 1$

$$\alpha = \frac{1}{p}$$

For $i = 0, \dots, N - 1$

$$y_{i+1} = x_i - \frac{1}{L} \nabla f(x_i)$$

$$\theta_{i+1} = \alpha + (1 - \alpha) \cdot \frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$$

$$x_{i+1} = y_{i+1} + \frac{\theta_i - 1}{\theta_{i+1}}(y_{i+1} - y_i)$$

9.1.4 Comparison between strategies

Since the only difference between the classical fast gradient method, the fast gradient method with fixed restart and these strategies is the evolution of the momentum parameter θ and, therefore, the evolution of the momentum coefficient $\beta_i = \frac{\theta_i - 1}{\theta_{i+1}}$, we can compare their evolution according to the number of iterations: Figure 9.1 displays the comparison for a value of $k = p = 10$. Note that on Figure 9.1a, when there are two values at one iteration, it corresponds to a change of value of θ at the same iteration: for example, for FGM-FR, if there is a restart at iteration $i + 1$, θ_{i+1} is first set to $\frac{1 + \sqrt{4\theta_i^2 + 1}}{2}$ and then it is set to 1 at the same iteration.

First, we can observe that for FGM-SR1, FGM-SR2a/b and FGM-FR, the evolution of θ and β are periodic of period k : this is logical since, in these algorithms, θ is set to 1 every k iterations.

Moreover, the evolution of the parameters θ is quite similar for FGM-SR1, FGM-SR2b and FGM-FR. The parameter β is also close for FGM-FR and FGM-SR1 but for FGM-SR2b, it explodes after each «full» restart: this can be explained by the fact that θ_{i+1} is set to 1 before the computation of β_i contrary to FGM-FR and FGM-SR1 where θ_{i+1} is set to 1 after the computation of β_i . Therefore, the value of β_i for FGM-SR2b is much greater than the one for FGM-FR and FGM-SR1.

Afterwards, for FGM-SR2a, even if the parameter θ increases much less than for the other strategies, the parameter β is quite similar at the beginning to the one obtained with FGM-FR and then, it starts to decrease more smoothly a few iterations before a «full» restart is achieved.

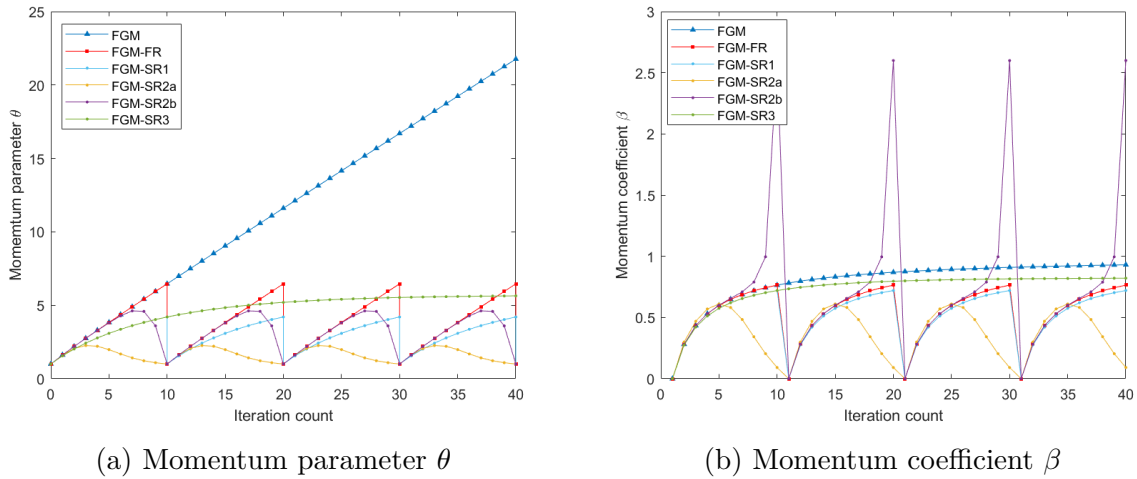


Figure 9.1: Comparison of the evolution of the momentum parameter θ and the momentum coefficient β for FGM, FGM-FR and the three strategies of FGM-SR for $k = p = 10$.

Finally, we can observe that the evolution of the parameter β is similar for FGM and FGM-SR3 even if the evolution of θ is completely different for these two methods: for FGM-SR3, θ increases at the beginning and then, tends to a certain value while for FGM, θ is always increasing according to the number of iterations.

9.2 Worst-case performance analysis

In this section, using the toolbox PESTO, we first analyze the worst-case performance for the three strategies of soft restart with $f(x_N) - f(x_*)$ as performance criterion and compare it with the worst-case performance of FGM and FGM-FR with the optimal restating interval k_{opt} found in Chapter 7. Then, we analyze the worst-case performance for the two other criteria, i.e., for $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$, but only for the promising strategy(ies) that allows improving the performance of FGM-FR. As in Sections 7.2 and 8.4, we limit

ourselves to $N = 80$ iterations since more iterations would require much more time and the goal is to have a general idea of the performance obtained with the different strategies of soft restart. Then, we consider two values of μ , i.e., $\mu \in \{0.01, 0.05\}$, and the following values of restarting intervals k and restarting factors p : $k, p \in \{5, 10, 20, 30, 60, k_{opt}\}$ where k_{opt} corresponds to the optimal restarting interval found in Chapter 7.

First, we can observe in Figure 9.2 that the worst-case performance of FGM-SR1 seems to follow a similar pattern than FGM-FR. Moreover, for both values of μ , the restarting interval providing the best performance seems to be close to the optimal restarting interval k_{opt} for FGM-FR. This is not surprising since the evolution of the momentum coefficient β is quite close for these two methods, as mentioned above.

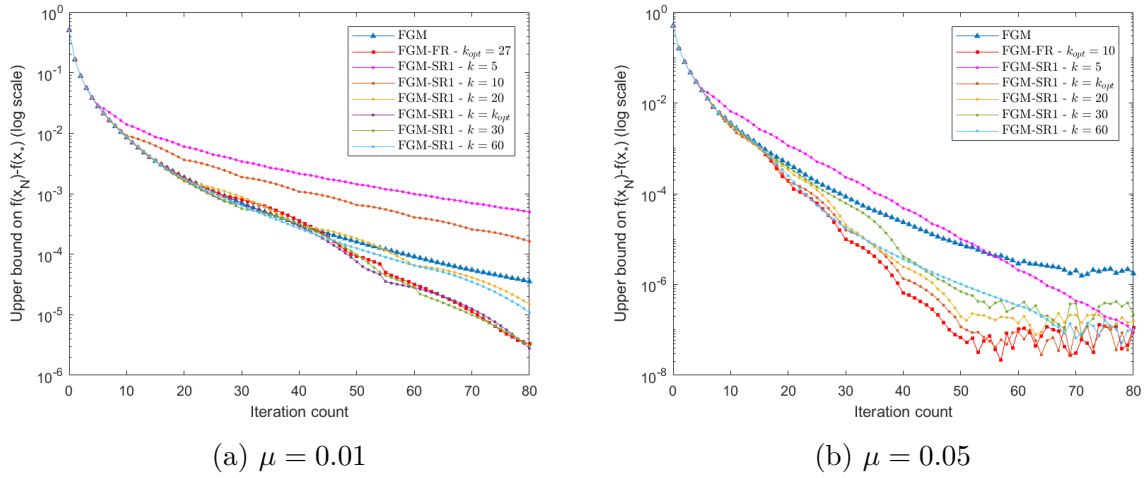


Figure 9.2: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR1 with several values of restarting interval k , for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

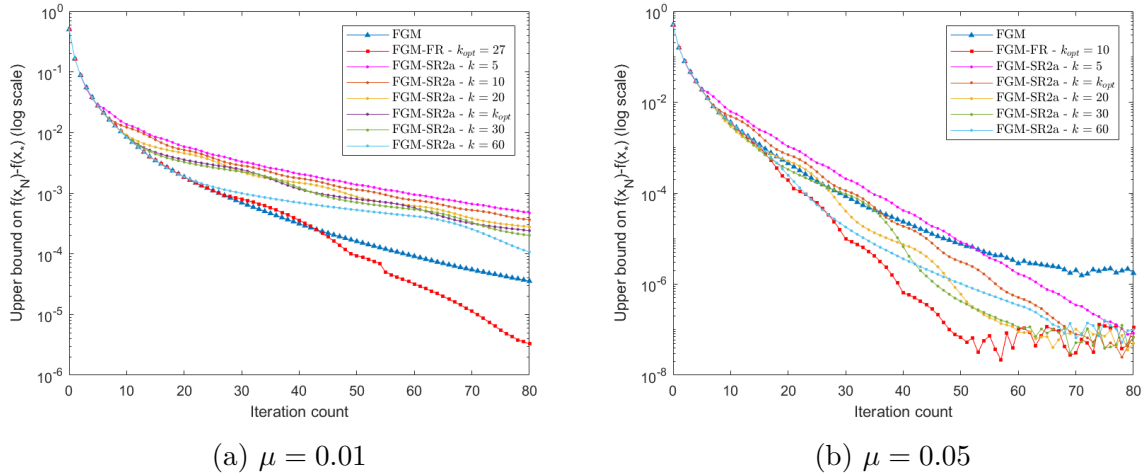


Figure 9.3: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR2a with several values of restarting interval k , for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

Then, Figure 9.3 shows that strategy 2a is worse than FGM-FR and even worse than the classical FGM for small values of μ . This could be due to the momentum coefficient β that decreases too quickly to 0 for this strategy.

Moreover, Figure 9.4 shows that depending on the value of k , the FGM-SR2b performs worse or only a bit better than the classical FGM: this can be explained by the fact that the momentum coefficient β explodes after each «full» restart, as mentioned above, which could be the cause of the oscillations observed in the performance evolution and therefore, which could considerably slow down the convergence.

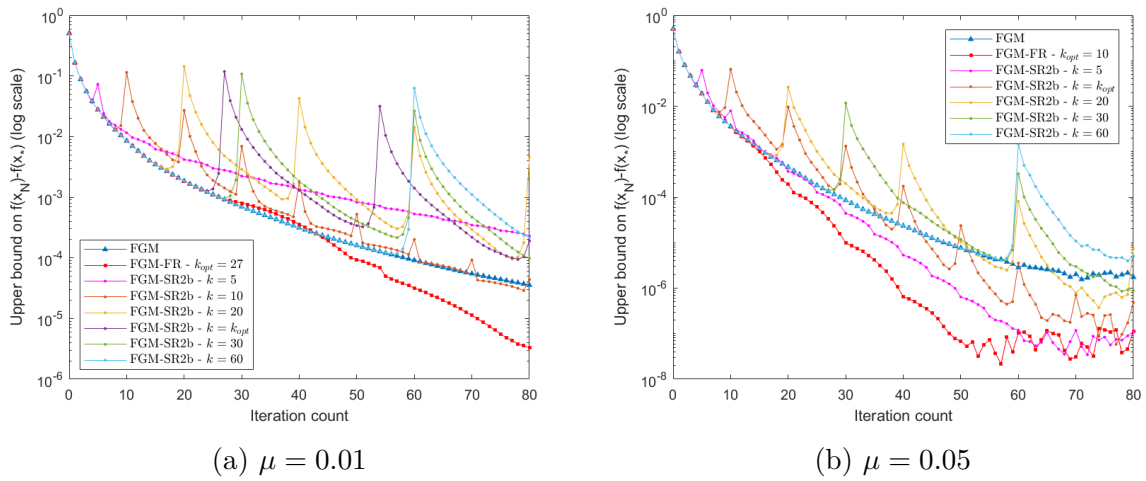


Figure 9.4: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR2b with several values of restarting interval k , for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

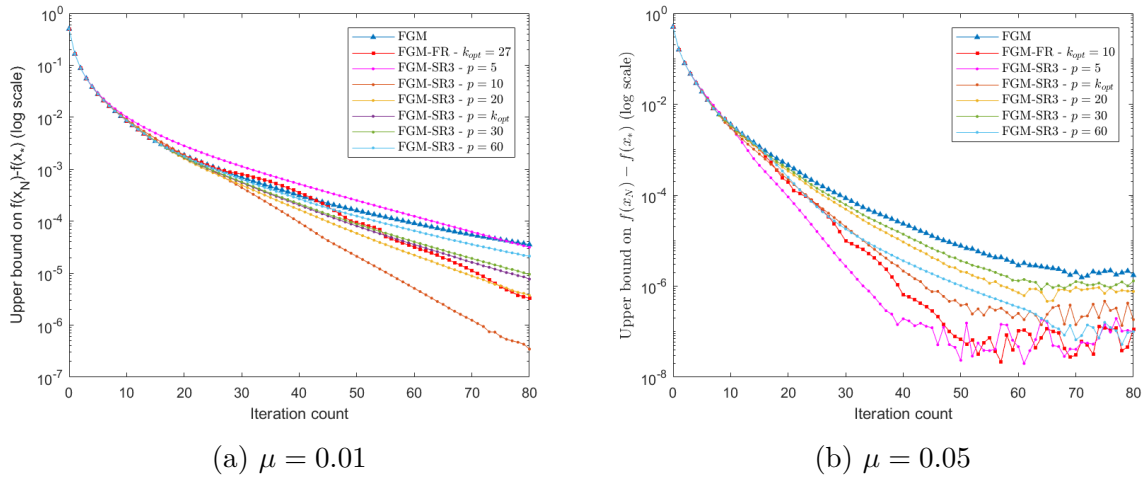


Figure 9.5: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR3 with several values of restarting factor p , for 1-smooth μ -strongly convex functions and $f(x_N) - f(x_*)$ as performance criterion.

Finally, we can see on Figure 9.5 that the third strategy performs better than FGM and FGM-FR for a specific value of p . Moreover, as for FGM-FR, it seems that there is an optimal parameter p_{opt} depending on the value of μ : indeed, for $\mu = 0.01$ and $\mu = 0.05$, the optimal p_{opt} seems to be respectively around 10 and around 5. Therefore, the optimal p_{opt} seems to be inversely proportional to μ such as k_{opt} for FGM-FR and it could be interesting to analyze if there is any relation between p_{opt} and k_{opt} .

Since the first two strategies do not seem promising but the third one allows to improve the performance of the fast gradient method with fixed restart, we have computed the worst-case performance for the two other performance criteria only for FGM-SR3 for the same values of μ and p .

In a general way, for all performance criteria considered, Figures 9.5, 9.6 and 9.7 show that FGM-SR3 allows to improve significantly the performance of FGM-FR when p is well chosen, especially for the criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. For example, after 80 iterations, with $f(x_N) - f(x_*)$ as performance criterion and for $\mu = 0.01$, the performance of FGM-SR3 is about 10 times better than the performance of FGM-FR (see Figure 9.5a).

Afterwards, we can also observe that the soft restart of strategy 3 does not create rebounds or oscillations in the evolution of the performance contrary to the fixed restart, which allows improving considerably the convergence of the FGM-SR3. This seems logical since we do not apply any «full » restart in this strategy and, therefore, no rebound appears.

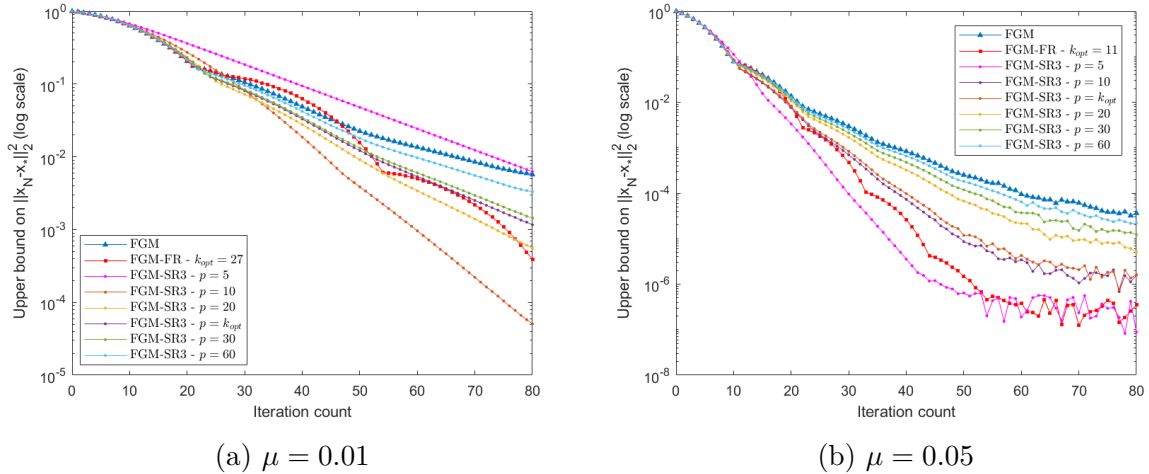


Figure 9.6: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR3 with several values of restarting factor p , for 1-smooth μ -strongly convex functions and $\|x_N - x_*\|_2^2$ as performance criterion.

Finally, unlike the optimal restarting interval k_{opt} , the optimal parameter p_{opt} seems to be around the same value for a specific value of μ and for all the performance criteria: for example, p_{opt} seems to be close to 10 for $\mu = 0.01$ and close to 5 for $\mu = 0.05$ for the three performance criteria considered. An interesting approach would therefore be to explore if the optimal parameter p_{opt} is close according to μ for all the performance criteria.

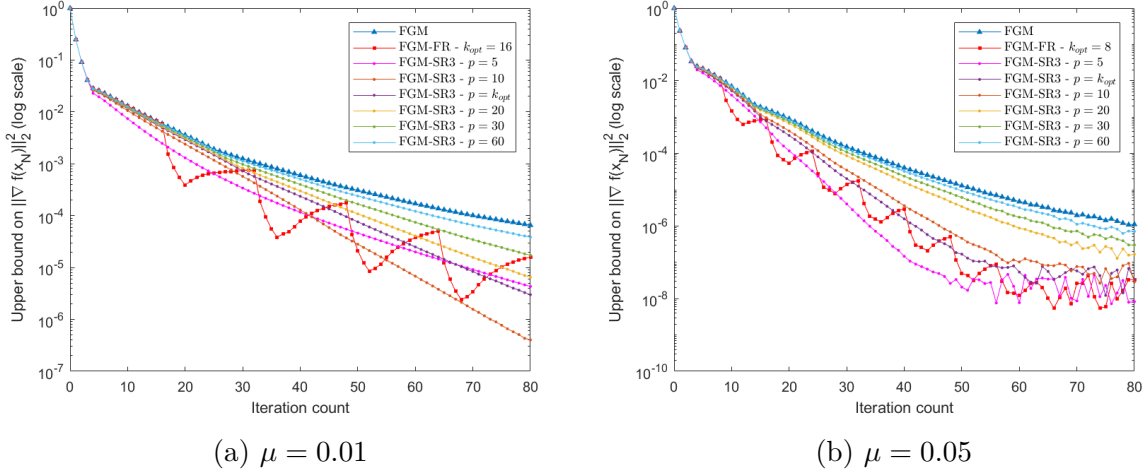


Figure 9.7: Comparison between the worst-case performance of FGM, FGM-FR with $k = k_{opt}$ and FGM-SR3 with several values of restarting factor p , for 1-smooth μ -strongly convex functions and $\|\nabla f(x_N)\|_2^2$ as performance criterion.

9.3 Optimal restarting factor

In this section, we implement an automatic strategy based on a specific criterion to determine the optimal restarting factor. In order to easily compare the optimal restarting interval k_{opt} for FGM-FR and the optimal restarting factor p_{opt} for FGM-SR3, we use the same strategy to compute p_{opt} and k_{opt} , as explained in Section 7.4. More precisely, we consider the following criterion:

Definition 9.1 (Criterion $N_{\mu,\epsilon}(p)$). The *criterion* $N_{\mu,\epsilon}(p)$ is the smallest number of iterations needed for FGM-SR3 with the restarting factor p to reach a specific precision ϵ on the worst-case performance for 1-smooth μ -strongly convex functions.

Then, the best restarting factor p_{opt} corresponds to the restarting factor p that minimizes $N_{\mu,\epsilon}(p)$ for a specific value of μ and precision ϵ :

$$p_{opt}(\mu, \epsilon) = \arg \min_p N_{\mu,\epsilon}(p).$$

In practice, as for the criterion $N_{\mu,\epsilon}(k)$, to compute the criterion $N_{\mu,\epsilon}(p)$ for given values of μ , ϵ and p and for a specific performance criterion, the idea is to evaluate the worst-case performance of FGM-SR3 for 1-smooth μ -strongly convex functions with the restarting factor p for every number of iterations from 1 until the performance is less or equal than ϵ : the last number of iterations considered corresponds then to $N_{\mu,\epsilon}(p)$.

Moreover, as for FGM-FR with the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, since the performance of FGM-SR3 is decreasing monotonically according to the number of iterations N for the three performance criteria considered, we compute the precision reached by FGM-SR3 every 10 iterations until the desired precision is reached. Therefore, the exact number of iterations $N_{\mu,\epsilon}(p)$ is either this iteration or one of the nine previous

ones and to find it, we inspect every iteration in this interval until the desired accuracy is reached.

Then, we apply the same strategy as the one explained in Section 7.4. Particularly, we consider the same values of precision ϵ depending on μ , the same initial searching space but in this case for the restarting factors p and we also limit the number of iterations to 15% above the current minimum number of iterations.

9.3.1 Analytical results

In this section, we apply the above strategy to determine the optimal restarting factor p_{opt} for several values of μ and the three performance criteria considered in this master thesis. For all performance criteria, we take the same range of values of μ : $\mu \in [0.2, 0.002]$ since this range is sufficiently large to obtain a general idea of the evolution of p_{opt} according to μ . Table 9.1 gives the results of the optimal restarting factor p_{opt} and displays the optimal restarting interval k_{opt} for all performance criteria ¹.

Table 9.1: Comparison between the optimal restarting interval k_{opt} and the optimal restarting factor p_{opt} for the three performance criteria considered and several values of μ ¹.

(a) $f(x_n) - f(x_*)$			(b) $\ x_n - x_*\ _2^2$			(c) $\ \nabla f(x_N)\ _2^2$		
μ	k_{opt}	p_{opt}	μ	k_{opt}	p_{opt}	μ	k_{opt}	p_{opt}
0.2	5	2	0.2	5	2	0.2	4-5	2
0.17	5	3	0.17	5-6	3	0.17	5	2
0.15	5-6	3	0.15	6	3	0.15	6	2-3
0.12	6-7	3	0.12	7	3	0.12	7	3
0.1	7	3	0.1	7-8	3	0.1	6-8	3
0.07	9	4	0.07	9-10	4	0.07	7	3-4
0.05	10	4-5	0.05	11-12	4-5	0.05	8	4
0.02	17	7	0.02	19-20	7	0.02	12	7
0.01	24-27	10	0.01	27-28	10	0.01	16	9-10
0.007	28	12	0.007	32	11-12	0.007	19	11-12
0.005	33-39	14	0.005	39	14-15	0.005	21	14
0.002	61	23	0.002	58-59	24	0.002	16-20	22

First, we can observe that the optimal restarting factor p_{opt} is almost the same for the three performance criteria and this holds for all values of μ . This observation is rather interesting since it means that choosing p as p_{opt} of a certain performance criterion minimizes the number of iterations needed to reach a certain accuracy for all the performance criteria.

Moreover, for the performance criterion $\|\nabla f(x_N)\|_2^2$, there is clearly no relation between k_{opt} and p_{opt} . However, for the two other performance criteria, both k_{opt} and p_{opt} increase

¹ After an update of the server, memory problems appeared and therefore, we considered $N_{max} = 150$ instead of $N_{max} = 200$. The results obtained with $N_{max} = 150$ are marked in red in the following tables.

as μ decreases but k_{opt} increases much faster. There does not seem to be an exact linear relationship between these two parameters but the optimal restarting factor p_{opt} seems to belong to a specific range of values $p_{opt} \in \left[\frac{k_{opt}}{3}, \frac{k_{opt}}{2}\right]$.

Then, since p_{opt} increases as μ decreases in the same way for all the performance criteria, it seems that the same formula can be found to approximate p_{opt} for all performance criteria. This will be done in the next section.

9.4 Optimal restarting factor formula

In the previous subsection, we observed that the optimal restarting factor p_{opt} for FGM-SR3 seems to be the same for all performance criteria. Therefore, it would be relevant to determine an optimal formula that fits with the optimal restarting factors found in Tables 9.1a, 9.1b and 9.1c.

As previously mentioned, the optimal restarting factor p_{opt} seems to belong to the range of values $p_{opt} \in \left[\frac{k_{opt}}{3}, \frac{k_{opt}}{2}\right]$. Thus, we have tried to find a multiplicative factor between $\frac{1}{3}$ and $\frac{1}{2}$ that fits with the optimal restarting factors p_{opt} when multiplied to the formula $e\sqrt{\frac{L}{\mu}}$. By choosing $\frac{3}{8}$ as multiplicative factor, the formula $\hat{p}_{opt} = \text{round}\left(\frac{3}{8}e\sqrt{\frac{L}{\mu}}\right)$ seems to match almost perfectly with the optimal restarting factor p_{opt} for a large range of values of μ and for the three performance criteria. Figure 7.5 displays the function $\hat{p}_{opt}$ and compares it to the optimal restarting factor p_{opt} for all performance criteria.

Therefore, the formula to approximate the optimal restarting factor is $\hat{p}_{opt} = \text{round}\left(\frac{3}{8}e\sqrt{\frac{L}{\mu}}\right)$ for all the performance criteria considered.

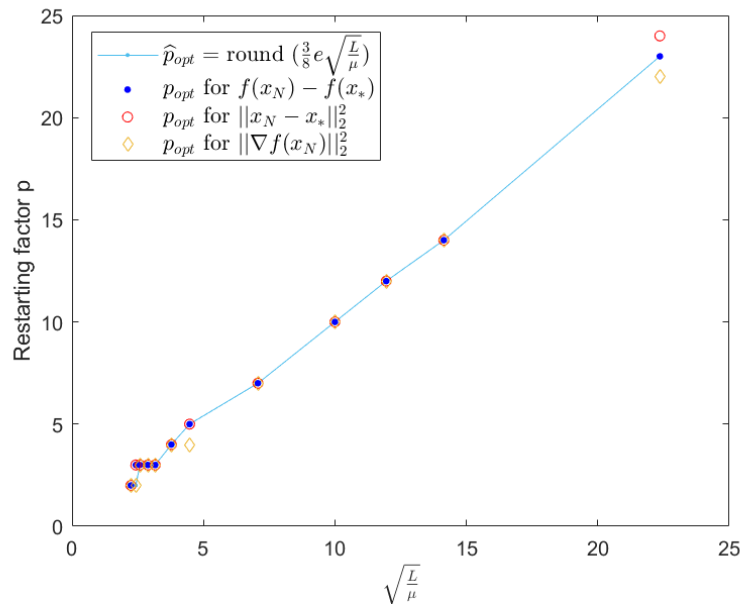


Figure 9.8: Comparison between the optimal restarting factor p_{opt} for the three performance criteria considered and the optimal restarting factor formula $\hat{p}_{opt}$.

9.5 Gain

To determine whether FGM-SR3 allows to really improve the worst-case performance of FGM and FGM-FR, we compute and compare the criterion $N_{\mu,\epsilon}(k)$ for $k \in \{k_{opt}, \hat{k}_{opt}, k_{max}\}$ and the criterion $N_{\mu,\epsilon}(p)$ for $p \in \{p_{opt}, \hat{p}_{opt}\}$ for the same values of μ as in Subsection 9.3.1 and for each performance criterion. Note that FGM-FR with $k = k_{max}$ corresponds to the classical FGM since no restart is applied for this restarting interval. To this end, the precision is adjusted according to the values of μ as in Table 7.1.

Tables 9.2, 9.3, 9.4 display the results and Figures 9.9, 9.10, 9.11 plot the results only for the same precision $\epsilon = 10^{-5}$ in order to display comparable results. Thus, for the performance criterion $\|x_N - x_*\|_2^2$, Figure 9.10 displays the results only for $\mu \geq 0.01$. Moreover, as previously explained in Section 7.5, since we consider a maximum number of iterations N_{max} , $N_{\mu,\epsilon}(k)$ can be bigger than N_{max} for some values of μ when $k = k_{max}$. Therefore, on Figures 9.9, 9.10 and 9.11, the points are unfilled when $N_{\mu,\epsilon}(k) > N_{max}$.

Then, as in Section 7.5, we analyze the relative gain depending on the choice of the fixed restarting interval $k = \{k_{opt}, \hat{k}_{opt}, k_{max}\}$ and/or the choice of the restarting factor $p = \{p_{opt}, \hat{p}_{opt}\}$. The gain is calculated as the difference between the number of iterations required to reach some precision for s_1 and s_2 , corresponding either to a restarting interval or a restarting factor:

$$G_{\mu,\epsilon}(s_1, s_2) = \frac{N_{\mu,\epsilon}(s_2) - N_{\mu,\epsilon}(s_1)}{N_{\mu,\epsilon}(s_2)}.$$

The relative gain gives the percentage of iterations we gain to reach a certain precision if we use s_1 instead of s_2 . This formula can only be used when $N_{\mu,\epsilon}(s)$ is known exactly.

Table 9.2: Comparison of $N_{\mu,10^{-5}}(k)$ and $N_{\mu,10^{-5}}(p)$ for different values of μ , k and p with $f(x_N) - f(x_*)$ as performance criterion.

μ	k_{opt}	$\hat{k}_{opt}$	p_{opt}	$\hat{p}_{opt}$	k_{max}
0.2	13	14	12	12	19
0.17	14	17	14	15	22
0.15	16	17	15	15	24
0.12	18	20	17	17	28
0.1	20	22	18	18	31
0.07	25	26	22	22	39
0.05	30	32	27	27	48
0.02	50	51	41	41	80
0.01	71	71	56	56	116
0.007	84	85	65	65	135
0.005	98	99	74	74	156
0.002	143	143	107	107	198

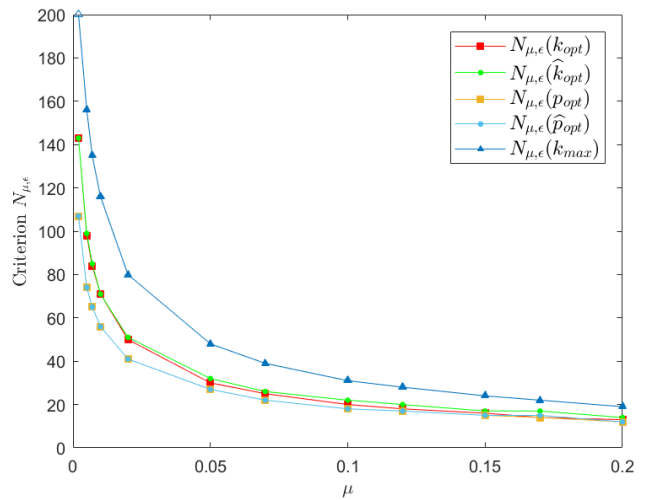


Figure 9.9: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ and $N_{\mu,10^{-5}}(p)$ according to μ for different values of k and p with $f(x_N) - f(x_*)$ as performance criterion.

Table 9.3: Comparison of $N_{\mu,\epsilon}(k)$ and $N_{\mu,\epsilon}(p)$ for different values of μ , k and p with $\|x_N - x_*\|_2^2$ as performance criterion [\[1\]](#).

μ	k_{opt}	$\hat{k}_{opt}$	p_{opt}	$\hat{p}_{opt}$	k_{max}
0.2	15	17	15	15	25
0.17	18	19	17	18	30
0.15	19	20	19	19	34
0.12	23	24	21	21	40
0.1	27	29	23	23	53
0.07	35	35	30	30	87
0.05	43	44	37	37	> 200
0.02	76	76	62	62	> 200
0.01	122	122	94	94	> 200
0.007	136	142	108	108	> 200
0.005	116	124	95	95	> 150
0.002	116	118	93	95	> 150

Table 9.4: Comparison of $N_{\mu,10^{-5}}(k)$ and $N_{\mu,10^{-5}}(p)$ for different values of μ , k and p with $\|\nabla(x_N)\|_2^2$ as performance criterion.

μ	k_{opt}	p_{opt}	$\hat{p}_{opt}$	k_{max}
0.2	13	12	12	21
0.17	13	14	14	23
0.15	15	15	15	25
0.12	17	17	17	30
0.1	20	18	18	34
0.07	23	23	23	43
0.05	27	27	28	53
0.02	40	42	42	92
0.01	52	58	58	134
0.007	61	67	67	163
0.005	67	77	77	193
0.002	84	112	113	>200

Firstly, compared to the classical fast gradient method, that is FGM-FR with $k = k_{max}$, FGM-SR3 with the optimal restarting factor p_{opt} improves significantly the worst-case performance for all values of μ considered: the relative gain $G_{\mu,\epsilon}(p_{opt}, k_{max})$ increases from 37% to 53% as μ decreases for the performance criterion $f(x_N) - f(x_*)$. For the performance criteria $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$, the relative gain is even bigger: it increases respectively from 40% to 65% and from 43% to 60% as μ decreases. Indeed, on Figures

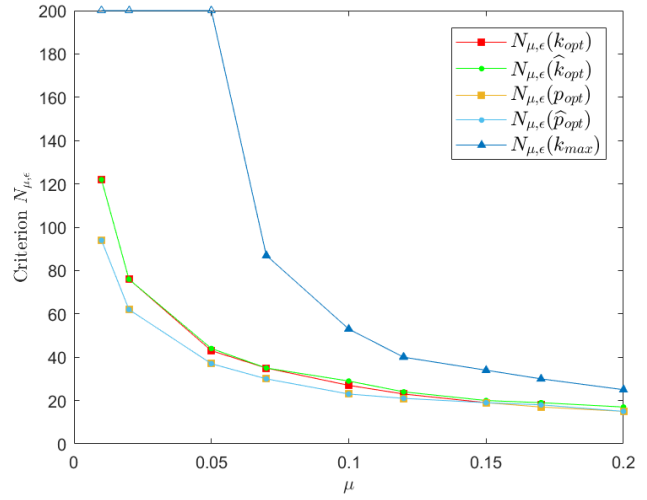


Figure 9.10: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ and $N_{\mu,10^{-5}}(p)$ according to μ for different values of k and p with $\|x_N - x_*\|_2^2$ as performance criterion.

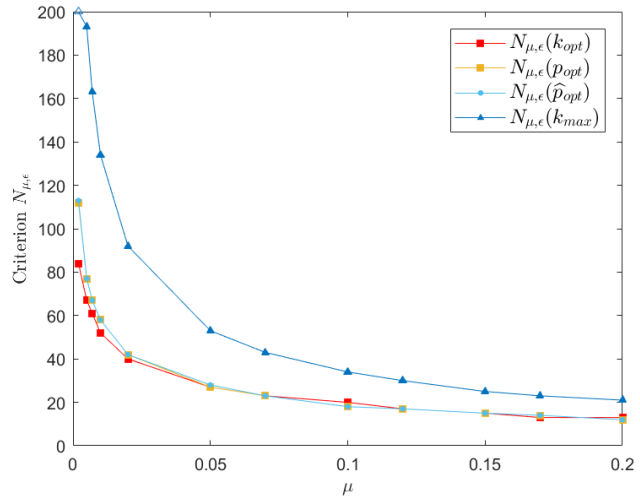


Figure 9.11: Evolution of the criterion $N_{\mu,10^{-5}}(k)$ and $N_{\mu,10^{-5}}(p)$ according to μ for different values of k and p with $\|\nabla f(x_N)\|_2^2$ as performance criterion.

[9.9](#), [9.10](#) and [9.11](#), we can see that the relative difference between $N_{\mu,\epsilon}(p_{opt})$, i.e., the yellow line, and $N_{\mu,\epsilon}(k_{max})$, i.e., the dark blue line, increases as μ decreases. Therefore, we can conclude that the restarting factor p_{opt} helps to improve significantly the classical fast gradient method.

Moreover, Tables [9.2](#), [9.3](#) and [9.4](#) show that the criteria $N_{\mu,\epsilon}(p_{opt})$ and $N_{\mu,\epsilon}(\hat{p}_{opt})$ are highly similar for all values of μ and for all performance criteria. This is logical since the optimal restarting factor formula $\hat{p}_{opt}$ approximates almost perfectly the optimal restarting factor p_{opt} . Therefore, using $\hat{p}_{opt}$ allows to considerably improve the performance of the classical fast gradient method: indeed, the relative gain $G_{\mu,\epsilon}(\hat{p}_{opt}, k_{max})$ is mainly the same as $G_{\mu,\epsilon}(p_{opt}, k_{max})$.

Afterwards, the worst-case performance of FGM-SR3 with $p = p_{opt}$ requires a lower number of iterations to reach a specific precision than FGM-FR with $k = k_{opt}$ but surprisingly, only for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. Indeed, on the one hand, the relative gain $G_{\mu,\epsilon}(p_{opt}, k_{opt})$ increases from 6% to 25% for $f(x_N) - f(x_*)$ and from 6% to 23% for $\|x_N - x_*\|_2^2$ as μ decreases. On the other hand, for the performance criterion $\|\nabla f(x_N)\|_2^2$, the criteria $N_{\mu,\epsilon}(k_{opt})$ and $N_{\mu,\epsilon}(p_{opt})$ are close for the biggest values of μ , i.e., $\mu \geq 0.05$. For example, $N_{\mu,\epsilon}(p_{opt}) = 12 < N_{\mu,\epsilon}(k_{opt}) = 13$ for $\mu = 0.2$ while $N_{\mu,\epsilon}(k_{opt}) = 13 < N_{\mu,\epsilon}(p_{opt}) = 14$ when $\mu = 0.17$ (see Table [9.4](#)). However, for the smallest values of μ , the criterion $N_{\mu,\epsilon}(k_{opt})$ is always lower than $N_{\mu,\epsilon}(p_{opt})$ and, therefore, FGM-FR with the optimal restarting interval k_{opt} better improves the worst-case performance than FGM-SR3 with the optimal restarting factor p_{opt} . Indeed, the relative gain $G_{\mu,\epsilon}(k_{opt}, p_{opt})$ increases from 5% to 25% as μ decreases from 0.05 to 0.002.

Finally, by computing FGM-SR3 with the restarting factor $\hat{p}_{opt}$ instead of FGM-FR with the restarting interval $\hat{k}_{opt}$, the relative gain $G_{\mu,\epsilon}(\hat{k}_{opt}, \hat{p}_{opt})$ increases respectively from 14% to 25% and from 12% to 24% for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ as μ decreases. This means that FGM-SR3 with $p = \hat{p}_{opt}$ further improves the worst-case performance of the classical FGM compared to FGM-FR with $k = \hat{k}_{opt}$.

9.6 Robustness

In the previous section, we have seen that FGM-SR3 with $p = \hat{p}_{opt}$ allows to significantly improve the performance of FGM and FGM-FR with $k = \hat{k}_{opt}$. Then, as in Section [7.6](#), we would like to know if FGM-SR3 using $p = \hat{p}_{opt}$ is robust to a wrong estimate of the μ parameter. Therefore, in this section, we analyze and compare the robustness depending on the estimation of μ for FGM-SR3 using $\hat{p}_{opt} = \text{round}\left(\frac{3}{8}e\sqrt{\frac{L}{\mu}}\right)$ as restarting factor, FGM-FR using $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$ as restarting interval and FGM- μ .

Note that since we have found a formula to approximate the optimal restarting factor p_{opt} for the three performance criteria, we compare the robustness of FGM-SR3 with $p = \hat{p}_{opt}$ and FGM- μ for all performance criteria. However, we compare the robustness of FGM-SR3 with $p = \hat{p}_{opt}$ and FGM-FR with $k = \hat{k}_{opt}$ only for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ since there is no formula to approximate the optimal restarting interval k_{opt} for the performance criterion $\|\nabla f(x_N)\|_2^2$.

To this end, we apply the same method as in Section 7.6: we fix the number of iterations N and then, we compare the precision reached after this number of iterations. For the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, we choose the same number of iterations as in Section 7.6, i.e., $N = N_{\mu, \epsilon}(\hat{k}_{opt})$, and for the performance criterion $\|\nabla f(x_N)\|_2^2$, we consider the number of iterations $N = N_{\mu, \epsilon}(\hat{p}_{opt})$. Indeed, these choices allow to obtain good performance without being too small to avoid numerical errors and, thus, a stagnation of the performance.

Then, the exact strong convexity parameter is denoted as μ_* and we consider the same under- and overestimation of μ_* as in Section 7.6:

$$\mu \in \left\{ \frac{\mu_*}{10}, \frac{\mu_*}{5}, \frac{\mu_*}{2}, \mu_*, 2\mu_*, 5\mu_*, 10\mu_* \right\}.$$

Figures 9.12, 9.13 and 9.14 show the comparison between the robustness of the methods FGM-FR, FGM- μ and FGM-SR3.

First, in a general way, we can observe that FGM-SR3 always gives the best performance when μ_* is known exactly for all values of μ_* tested and all performance criteria considered. For the performance criterion $\|x_N - x_*\|_2^2$, the performance of FGM- μ and FGM-SR3 are really close when $\mu = \mu_*$ (see Figure 9.13) but for the two other performance criteria, the performance of FGM-SR3 becomes better than the performance of FGM- μ as μ_* decreases (see Figures 9.12 and 9.14). For example, for the performance criterion $f(x_N) - f(x_*)$, when $\mu_* = 0.1$, the performance of FGM- μ and FGM-SR3 are almost equal but when $\mu_* = 0.01$, the performance of FGM-SR3 is about 12 times better than the performance of FGM- μ . These results are quite surprising since FGM- μ is a method specifically designed for smooth strongly convex functions.

Moreover, for underestimations of μ_* , FGM-SR3 is much more robust than FGM- μ but is less robust than FGM-FR. For example, for $\mu_* = 0.01$, for the performance criterion $f(x_N) - f(x_*)$ and for FGM-FR, taking $\mu = \frac{\mu_*}{10}$ gives a performance that is only about 5 times worse than with the exact value of μ_* while for FGM-SR3, taking $\mu = \frac{\mu_*}{10}$ gives a performance that is about 18 times worse than with the exact value of μ_* (see Figure 9.12a). However, the performance of FGM-SR3 is better than the performance of FGM-FR for all underestimations of μ_* , i.e., for all $\mu < \mu_*$, and for all performance criteria. This can be explained by the fact that the performance of FGM-SR3 is much better than the performance of FGM-FR when μ_* is known exactly. Therefore, even if FGM-SR3 suffers from a bigger relative performance loss than FGM-FR when μ_* is underestimated, FGM-SR3 gives still better performance. This is quite interesting since it allows users to use conservative estimates of μ_* without too much impact on the performance. Then, since FGM-SR3 is robust to underestimation, it means that this method will take advantage if the function is better conditioned around the minimum. Indeed, as explained in Section 7.6 for FGM-FR, if the function is better conditioned around the minimum, it amounts to underestimate locally the parameter μ_* around the minimum.

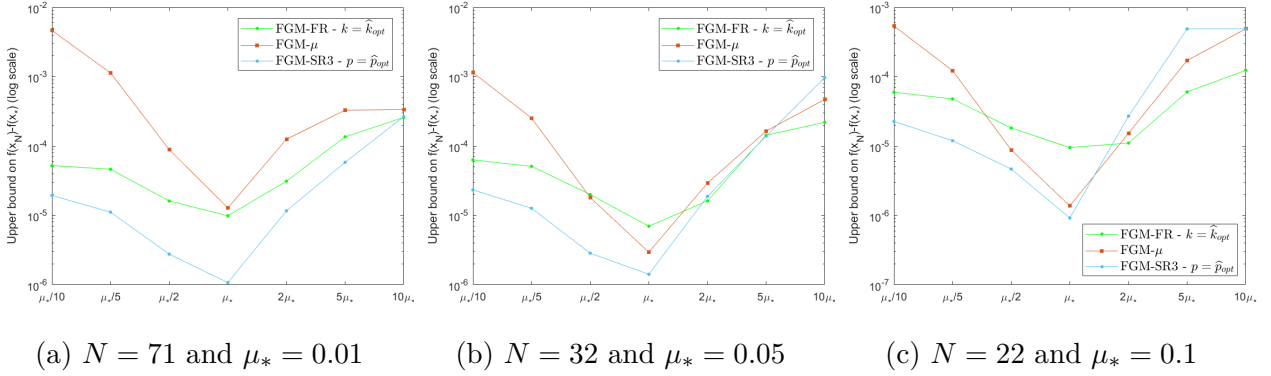


Figure 9.12: Comparison of the performance of FGM-FR with $k = \hat{k}_{opt}$, FGM- μ and FGM-SR3 with $p = \hat{p}_{opt}$ after $N = N_{\mu, \epsilon}(\hat{k}_{opt})$ iterations for 1-smooth μ_* -strongly convex functions for several estimations of μ_* and $f(x_N) - f(x_*)$ as performance criterion.

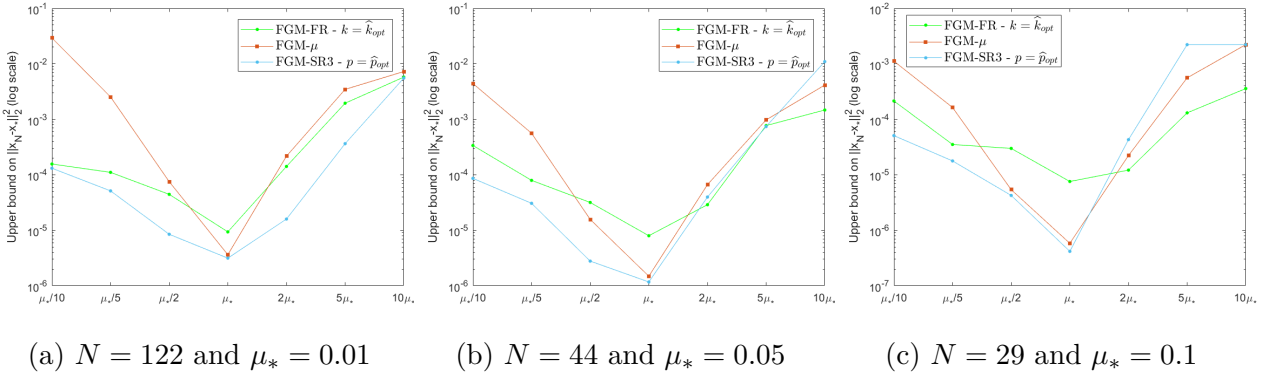


Figure 9.13: Comparison of the performance of FGM-FR with $k = \hat{k}_{opt}$, FGM- μ and FGM-SR3 with $p = \hat{p}_{opt}$ after $N = N_{\mu, \epsilon}(\hat{k}_{opt})$ iterations for 1-smooth μ_* -strongly convex functions for several estimations of μ_* and $\|x_N - x_*\|_2^2$ as performance criterion.

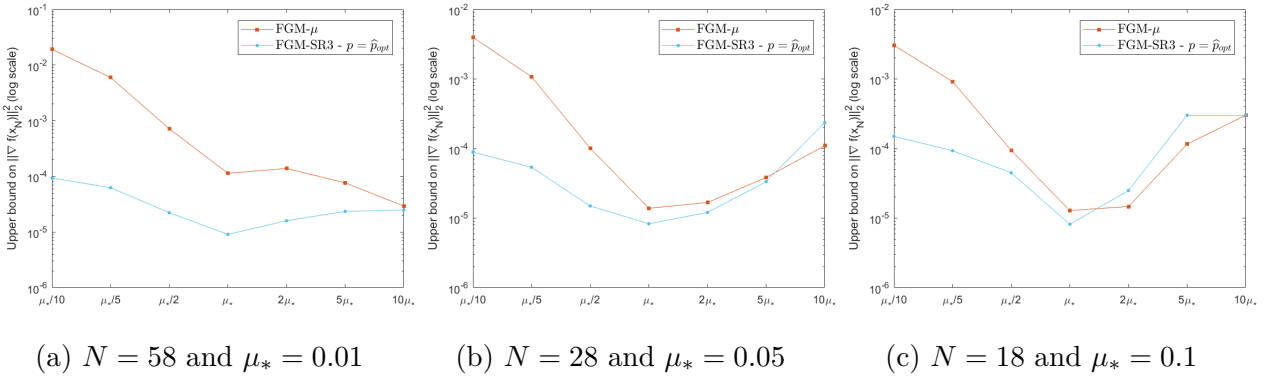


Figure 9.14: Comparison of the performance of FGM- μ and FGM-SR3 with $p = \hat{p}_{opt}$ after $N = N_{\mu, \epsilon}(\hat{p}_{opt})$ iterations for 1-smooth μ_* -strongly convex functions for several estimations of μ_* and $\|\nabla f(x_N)\|_2^2$ as performance criterion.

Afterwards, overestimating μ_* does not have the same impact on the performance according to the performance criterion considered.

On the one hand, for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, FGM-SR3 is much less robust to overestimation of μ_* than to underestimation, especially for high values of μ_* . For example, for the performance criterion $\|x_N - x_*\|_2^2$, when $\mu_* = 0.01$, the performance obtained with FGM-SR3 for $\mu = 5\mu_*$ is about 200 times worse than with the exact value of μ_* (see Figure 9.13a) while when $\mu_* = 0.1$, the performance obtained with FGM-SR3 for $\mu = 5\mu_*$ is about 5000 times worse than with the exact value of μ_* (see Figure 9.13c). Therefore, for high values of μ_* , i.e., $\mu_* \geq 0.05$, overestimating the parameter μ_* gives worse performance for FGM-SR3 than for FGM- μ and FGM-FR. However, for small μ_* , FGM-SR3 gives better performance for overestimates of μ_* than for FGM- μ and FGM-FR.

On the other hand, for the performance criterion $\|\nabla f(x_N)\|_2^2$, FGM-SR3 seems to be more robust to overestimate μ_* , especially when μ_* is small. For example, taking $\mu = 5\mu_*$ gives a performance that is only about 2 times worse than with the exact value of μ_* when $\mu_* = 0.01$ (see Figure 9.14a).

Then, we can see on Figures 9.12c, 9.13c and 9.14c that when $\mu_* = 0.1$, the performance of FGM-SR3 for $\mu = 5\mu_*$ and $\mu = 10\mu_*$ are equal: this is due to the fact that $\hat{p}_{opt} = \text{round}\left(\frac{3}{8}e\sqrt{\frac{L}{\mu}}\right)$ is a rounded formula and is therefore the same for both these values of μ .

Finally, for the performance criterion $\|\nabla f(x_N)\|_2^2$, FGM- μ is more robust to overestimation of the parameter μ_* than for the two other performance criteria. For $\mu_* = 0.01$, we can observe on Figure 9.14a that overestimating μ_* for FGM- μ gives better performance than by taking the exact value of μ_* . This can be explained by the fact that the number of iterations N is too small and after this number of iterations, FGM- μ_* is not yet better than FGM- μ with $\mu \geq 5\mu_*$.

To conclude, in a general way, even if FGM-SR3 is less robust to wrong estimations of μ_* than FGM-FR, it gives better performance than FGM-FR and FGM- μ when the exact strong convexity parameter μ_* is exact or underestimated. For overestimates of μ_* , FGM-SR3 is robust only for the performance criterion $\|\nabla f(x_N)\|_2^2$ but not for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$, especially for high values of μ_* . Therefore, for all performance criteria, it is better to use FGM-SR3 when the estimation of μ_* is exact or too conservative and for large overestimation of μ_* when μ_* is high, it is better to use FGM- μ for the performance criterion $\|\nabla f(x_N)\|_2^2$ and FGM-FR with $k = \hat{k}_{opt}$ for the two other performance criteria.

9.7 Conclusion

In this chapter, we have first seen that the two first strategies of soft restart improve the classical FGM for high values of μ but they do not improve FGM-FR. However, the fast gradient with soft restart using the third strategy FGM-SR3 allows to significantly improve the performance of FGM and FGM-FR for all performance criteria considered when the restarting factor p is well chosen.

Afterwards, we have evaluated the best restarting factor p_{opt} , i.e., the value of p which gives the best performance, for a large range of values of μ based on a specific criterion $N_{\mu,\epsilon}(p)$ for all performance criteria considered. Then, we have derived a formula $\hat{p}_{opt}$ depending on μ and L which allows to estimate p_{opt} for the three performance criteria: $\hat{p}_{opt} = \text{round} \left(\frac{3}{8} e \sqrt{\frac{L}{\mu}} \right)$.

Compared to the classical FGM, FGM-SR3 with $p = \hat{p}_{opt}$ reduces the number of iterations needed to reach a specific precision from respectively 37% to 53%, 40% to 65%, 43% to 60% as μ decreases for the performance criteria $f(x_N) - f(x_*)$, $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$. Moreover, compared to FGM-FR with $k = \hat{k}_{opt}$, the relative gain is improved from 14% to 25% and from 12% to 24% respectively for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$ as μ decreases. Those outcomes confirm that FGM-SR3 with $p = \hat{p}_{opt}$ significantly improves the worst-case performance of FGM and FGM-FR with $k = \hat{k}_{opt}$.

Finally, the robustness of FGM-SR3 with $p = \hat{p}_{opt}$ has been analyzed for under- or overestimates of the μ parameter and compared with the robustness of FGM- μ and FGM-FR with $k = \hat{k}_{opt}$. Even if FGM-SR3 is less robust than FGM-FR, it allows to improve significantly the performance of FGM-FR and FGM- μ for exact and underestimations of μ_* . However, for overestimations of μ_* , FGM-SR3 is robust only for the performance criterion $\|\nabla f(x_N)\|_2^2$ but still gives better performance for all performance criteria when μ_* is small.

Therefore, we have found a version of the fast gradient method which allows improving the performance of the classical FGM and FGM-FR for all performance criteria and which is robust for underestimations of the strong convexity parameter.

10

Conclusion

Recently, with the growth of computational capabilities, optimization methods have become a powerful tool for making decisions in many different areas such as medicine, manufacturing, biology, economy, etc. These methods have attracted much attention from researchers who have invested a lot of work to design or improve methods to solve optimization problems.

More formally, an optimization problem searches the *decision variables* x within a set $\mathcal{X}$ which minimize - or maximize - an *objective function* f . In daily life, almost every application can be modelled using this optimization framework but no strong guarantees for solving them are available. Therefore, several assumptions are added on both the function f and the decision space $\mathcal{X}$, which allows having strong guarantees on the solution obtained from specific algorithms.

Many methods allow solving this optimization problem. In this master thesis, we consider iterative black-box first-order optimization methods which only require evaluation of the objective function f and its gradient at a sequence of iterates. We focus on these methods since they proved to be successful to solve high-dimensional problem instances.

To compare these optimization methods and determine the most appropriate for solving specific optimization problems, one can evaluate the performance. More particularly, in this master thesis, we evaluate the worst-case performance for three performance criteria: the function value accuracy $f(x_N) - f(x_*)$, the distance to the optimal solution $\|x_N - x_*\|_2^2$ and the residual gradient norm $\|\nabla f(x_N)\|_2^2$.

Chapter 2 first defines the problem of finding the worst-case performance: given a class of problems and a first-order method, the idea is to calculate the worst-case accuracy of the solutions obtained after a given number of iterations for a specific performance criterion. Then, it describes and defines all the assumptions made throughout this master thesis to obtain strong guarantees on the worst-case performance: we consider an unconstrained optimization problem for the class of L -smooth μ -strongly convex functions, i.e., $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$. Finally, it presents the methods used and analyzed in this master thesis: the gradient method, i.e., GM, which moves the iterates by a fixed step size in the direction of the steepest slope at each iteration and the fast gradient method, i.e., FGM, which accelerates the gradient method by using a momentum parameter. For each method, performance bounds and convergence properties for smooth convex functions and smooth

strongly convex functions are given.

However, upper bounds on the worst-case performance are not necessarily exact and thus we cannot conclude using performance bounds which method works better in practice: indeed, some methods could converge much faster than their theoretical performance bound.

Therefore, to better evaluate the worst-case performance, we can use the toolbox PESTO developed by Adrien Taylor that computes the exact worst-case performance of fixed-step first-order optimization methods.

Chapter 3 presents the performance estimation problem that turns the problem of finding the exact worst-case performance of a black-box first-order optimization method into an optimization problem over a set of smooth (strongly) convex functions and some initial conditions. Then, it explains how this problem can be formulated as a convex semidefinite program that can be solved numerically for fixed-step first-order methods. This approach is implemented in the toolbox PESTO that provides a framework in which methods can be described quite easily. However, PESTO cannot deal with If conditions on variables that influence the step size of first-order methods or strict inequalities. Moreover, the time for solving the problem of finding the exact worst-case performance using PESTO increases exponentially with the number of iterations applied by the method considered.

Chapter 4 studies the convergence and the exact worst-case performance of the gradient and fast gradient method using the toolbox PESTO for smooth convex and smooth strongly convex functions. It turns out that, unlike GM, FGM derives little benefits from the strong convexity property. Therefore, the goal of this master thesis is to improve the performance of this method for smooth strongly convex functions.

Chapter 5 provides a preamble on accelerated first-order methods that have been introduced in the literature in the context of continuous unconstrained optimization. This chapter shows that several versions of FGM, such as FGM- μ , i.e., a version FGM for L -smooth μ -strongly convex functions which require parameters L and μ in its implementation, allow improving the performance of FGM for smooth strongly convex functions. However, under- or overestimating the momentum term can have a severe detrimental effect on the convergence of accelerated gradient schemes. Therefore, tuning accelerated gradient methods designed for smooth strongly convex functions based on wrong estimates of μ leads to a significant slow down of the convergence: these methods are thus non-robust to wrong estimations of the parameter μ .

Therefore, another goal of this master thesis is to determine versions of the fast gradient method that do not depend on the strong convexity parameter μ or methods that are robust even when the estimation of μ is not accurate. To this end, the restart scheme has been introduced in the literature: applying a restart consists of stopping the algorithm, erasing the memory and taking the last iterate as a new initial point.

Now that our goals are identified, **Chapter 6** explained the main contributions of our master thesis. We will implement and analyze three different restarting schemes: the fixed restart which consists of applying a restart after a fixed number of iterations, the adaptive restart which consists of applying a restart when a certain restarting condition is satisfied

and the soft restart which consists of applying a fraction restart at each iteration. The fixed restart and the adaptive restart have already been implemented in the literature to solve some specific problems while the soft restart is a completely new approach.

Chapter 7 first explains the fast gradient method with a fixed restart, i.e., FGM-FR: the idea is to restart periodically FGM after a fixed number of iterations k . In practice, it amounts to fixing the momentum parameter θ to 1 every k iterations. Then, we have analyzed the worst-case performance of FGM-FR for several values of k and we have observed that this method helps to improve the worst-case performance of the classical fast gradient method for strongly convex functions when the restarting interval k is well chosen.

Afterwards, Chapter 7 evaluates the best restarting interval k_{opt} , i.e., the value of k which gives the best worst-case performance, for a large range of values of μ based on a specific criterion for the three performance criteria considered. Then, we have derived a formula $\hat{k}_{opt}$ depending on L and μ that allows to estimate k_{opt} but only for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$: $\hat{k}_{opt} = \text{round}\left(e\sqrt{\frac{L}{\mu}}\right)$.

Compared to the classical FGM, FGM-FR with $k = \hat{k}_{opt}$ reduces the number of iterations needed to reach a specific precision by on average 32% for the performance criterion $f(x_N) - f(x_*)$ and from 32% to 60% as μ decreases for the performance criterion $\|x_N - x_*\|_2^2$, which is quite a significant improvement.

Finally, Chapter 7 analyzes and compares the robustness of FGM-FR with $k = \hat{k}_{opt}$ and FGM- μ for wrong estimations of the exact strong convexity parameter μ_* . FGM-FR with $k = \hat{k}_{opt}$ is robust when the estimation of the strong convexity parameter μ_* is not accurate, especially for large underestimations of the exact value of μ_* . Unfortunately, overestimating μ_* gives some performance losses for FGM-FR but not as many as for FGM- μ . It is nevertheless better to use FGM- μ when μ_* is known exactly.

Chapter 8 first describes the fast gradient method with adaptive restart, i.e., FGM-AR, with two restarting conditions: the function scheme and the gradient scheme. Then, this chapter explains how to implement FGM-AR using the toolbox PESTO by dealing with the restarting conditions which are strict inequalities and depend on the variables. We show that it is more complicated than expected: indeed, on the one hand, a δ parameter is used to remove the strict inequality but due to its influence on the worst-case performance, the problem of finding the worst-case performance of FGM-AR becomes ill-defined. On the other hand, since PESTO cannot deal with If conditions on variables, the computation of the worst-case performance of FGM-AR must be separated into several cases corresponding to every possible combinations of restart/no restart and the worst-case performance of FGM-AR is then given by the worst performance among the worst-case performance of all cases. Thus, the number of cases grows exponentially as the number of iterations increases or as the restarting interval k decreases.

To get rid of these issues, Chapter 8 computes the worst-case performance of a particular case: FGM-AR without restart, i.e., the case where no adaptive restart is triggered, which gives a lower bound on the worst-case performance of FGM-AR. Indeed, this case does not need to use a δ term since there is no strict inequality in this case and since we focus only

on one particular case, the worst-case performance can be computed for a large number of iterations with an adaptive restart every iteration, i.e., $k = 1$.

Chapter 8 then compares the worst-case performance of FGM-AR without restart and the classical FGM. Since the performance of both methods is close for both restarting conditions and all performance criteria considered, we can conclude that FGM-AR with these restarting conditions does not allow to improve the performance of FGM and is thus inefficient in terms of the worst-case.

Chapter 9 first explains the fast gradient method with soft restart, i.e., FGM-SR, and develops three different strategies of FGM-SR. The idea is to apply a fraction of restart at each iteration instead of applying a «full» restart after a constant number of iterations as the fixed restart does. To this end, we define a parameter $\alpha \in [0, 1]$, called the restarting fraction, that controls the evolution of the momentum parameter θ and the three strategies have a different restarting fraction α . Then, this chapter compares the worst-case performance depending on the choice of the soft restart strategy. The two first strategies slightly improve the classical fast gradient method for high values of μ but do not improve the fast gradient method with fixed restart. However, the third strategy of the fast gradient method with soft restart, i.e., FGM-SR3, allows to significantly improve the performance of FGM and FGM-FR for all performance criteria considered when the restarting factor p is well chosen. This strategy consists of applying a p^{th} of restart at each iteration by setting $\alpha = \frac{1}{p}$.

Afterwards, Chapter 9 evaluates the best restarting factor p_{opt} , i.e., the value of p which gives the best worst-case performance, for a large range of values of μ based on a specific criterion for the three performance criteria considered. Then, Chapter 9 derives a formula $\hat{p}_{opt}$ depending on L and μ that allows to estimate p_{opt} for the three performance criteria: $\hat{p}_{opt} = \text{round} \left(\frac{3}{8} e \sqrt{\frac{L}{\mu}} \right)$.

Compared to the classical FGM, FGM-SR3 with $\hat{p}_{opt}$ reduces the number of iterations needed to reach a specific precision from respectively 37% to 53%, 40% to 65%, 43% to 60% as μ decreases for the performance criteria $f(x_N) - f(x_*)$, $\|x_N - x_*\|_2^2$ and $\|\nabla f(x_N)\|_2^2$. Moreover, compared to FGM-FR with $k = \hat{k}_{opt}$, FGM-SR3 with $\hat{p}_{opt}$ reduces the number of iterations needed to reach a specific precision from 14% to 25% and from 12% to 24% as μ decreases respectively for the performance criteria $f(x_N) - f(x_*)$ and $\|x_N - x_*\|_2^2$. Those outcomes confirm that FGM-SR3 with $p = \hat{p}_{opt}$ significantly improves the worst-case performance of FGM and FGM-FR with $k = \hat{k}_{opt}$.

Finally, the robustness of FGM-SR3 with $p = \hat{p}_{opt}$ has been analyzed for under- or overestimates of the exact strong convexity parameter μ_* and compared with the robustness of FGM- μ and FGM-FR with $k = \hat{k}_{opt}$. Even if FGM-SR3 is less robust than FGM-FR, it allows to improve significantly the performance of FGM-FR and FGM- μ for exact and underestimations of μ_* . However, for overestimates of μ_* , FGM-SR3 is robust only for the performance criterion $\|\nabla f(x_N)\|_2^2$ but still gives better performance for all performance criteria when μ_* is small.

To conclude, in this master thesis, thanks to the toolbox PESTO, we were able to evaluate

and analyze the worst-case performance of accelerated gradient methods using a restart scheme. This allows us to derive two versions of the fast gradient method which improve the performance of the classical fast gradient method for all performance criteria: the fast gradient with fixed restart FGM-FR using an optimal restarting interval formula $\hat{k}_{opt}$ and the fast gradient with soft restart with a specific strategy FGM-SR3 using an optimal restarting factor formula $\hat{p}_{opt}$.

Further work can be applied to improve the performance of accelerated gradient methods using restart schemes. First, a further improvement would be to determine the optimal restarting interval k and the optimal restarting factor p for a larger range of values of the μ parameter, especially for smaller values of μ since the classical FGM already works very well and converges fast for high values of μ . This would allow having stronger certifications and ensuring the accuracy of our optimal restarting formulas $\hat{k}_{opt}$ and $\hat{p}_{opt}$. Then, to enhance the fast gradient method with adaptive restart, one could try to find other restarting conditions that would improve the worst-case performance. It would be relevant to start by comparing the worst-case performance of FGM-AR without restart with these restarting conditions and the classical FGM: if their performance is very close, we could directly conclude that these restarting conditions do not improve the performance of FGM. Moreover, further work could focus on the understanding and the analysis of soft restart from a theoretical point of view and on the reasons why FGM-AR does not improve the worst-case performance. Afterwards, one could try to find other strategies of soft restart that would further improve the worst-case performance of the fast gradient method. Finally, it would be interesting to compare FGM-FR and FGM-SR3 to new efficient gradient methods designed for smooth strongly convex functions such as the Triple Momentum Method [VFL18] or the Information-Theoretic Exact Method [TD21].

The study of optimization methods is a very active research area with many applications such as machine learning, artificial intelligence, image processing, etc. Through this master thesis, we believe that we have contributed to the study of first-order optimization methods, especially the fast gradient method, and helped to improve their performance. In our opinion, these methods have an essential role to play in our society where they are used in a multitude of concrete domains.

Bibliography

- [BB08] Léon Bottou and Olivier Bousquet. “The tradeoffs of large scale learning”. In: *Advances in Neural Information Processing Systems*. 2008, pp. 161–168.
- [BCG10] Stephen Becker, Emmanuel J. Candès, and Michael Grant. “Templates for convex cone problems with applications to sparse signal recovery”. In: *Math. Prog. Comp* (2010), pp. 165–218. URL: <https://arxiv.org/pdf/1009.2065.pdf>.
- [BT09] Amir Beck and Marc Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. URL: <http://members.cbio.mines-paristech.fr/~jvert/svn/bibli/local/Beck2009fast.pdf>.
- [CC19] Luca Calatroni and Antonin Chambolle. “Backtracking strategies for accelerated descent methods with smooth composite objectives”. In: *SIAM Journal on Optimization* 29.3 (2019), pp. 1772–1798. URL: <https://hal.archives-ouvertes.fr/hal-01596103>.
- [DT12] Yoel Drori and Marc Teboulle. “Performance of first-order methods for smooth convex minimization: A novel approach”. In: *Mathematical Programming* 145 (2012). DOI: [10.1007/s10107-013-0653-0](https://doi.org/10.1007/s10107-013-0653-0).
- [Faz+18] Mahyar Fazlyab, Alejandro Ribeiro, Manfred Morari, and Victor M. Preciado. *Analysis of Optimization Algorithms via Integral Quadratic Constraints: Nonstrongly Convex Problems*. 2018.
- [GB14] Pontus Giselsson and Stephen Boyd. “Monotonicity and restart in fast gradient methods”. In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 5058–5063. DOI: [10.1109/CDC.2014.7040179](https://doi.org/10.1109/CDC.2014.7040179).
- [Gli19a] François Glineur. *Slides from the course LINMA24171 "First-order methods: 2.1 Gradient method"*. 2019.
- [Gli19b] François Glineur. *Slides from the course LINMA24171 "First-order methods: 2.1 More about gradient method"*. 2019.
- [KF15] Donghwan Kim and Jeffrey A. Fessler. “Optimized first-order methods for smooth convex minimization”. In: *Mathematical Programming* 159.1-2 (2015), pp. 81–107. ISSN: 1436-4646. DOI: [10.1007/s10107-015-0949-3](https://doi.org/10.1007/s10107-015-0949-3).

- [KGT17] Etienne de Klerk, François Glineur, and Adrien B. Taylor. *On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions*. 2017.
- [LRP16] Laurent Lessard, Benjamin Recht, and Andrew Packard. *Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints*. 2016.
- [Nem94] Arkadi Nemirovski. *Efficient methods in convex programming*. 1994. URL: http://www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf.
- [Nes04] Yurii Nesterov. “Introductory Lectures on Convex Optimization - A Basic Course”. In: *Applied Optimization*. 2004.
- [Nes12] Yurii Nesterov. *A Lyapunov Analysis of Momentum Methods in Optimization*. 2012. URL: <http://www.mathopt.org/Optima-Issues/optima88.pdf>.
- [Nes13] Yurii Nesterov. “Gradient methods for minimizing composite functions”. In: *Math. Program.* 140 (2013), pp. 125–161. DOI: <https://doi.org/10.1007/s10107-012-0629-5>.
- [Nes18] Yurii Nesterov. *Lectures on Convex Optimization*. 2nd. Springer Publishing Company, Incorporated, 2018. ISBN: 3319915770.
- [Nes83] Yurii Nesterov. “A method for solving the convex programming problem with convergence rate $O(1/k^2)$ ”. In: *Proceedings of the USSR Academy of Sciences* 269 (1983), pp. 543–547.
- [NNG16] Ion Necoara, Yurii Nesterov, and François Glineur. *Linear convergence of first order methods for non-strongly convex optimization*. 2016. arXiv: [1504.06298](https://arxiv.org/abs/1504.06298) [math.OA].
- [OC15] Brendan O’Donoghue and Emmanuel Candès. “Adaptive Restart for Accelerated Gradient Schemes”. In: *Found. Comput. Math.* 15 (2015), pp. 715–732. DOI: <https://doi.org/10.1007/s10208-013-9150-3>.
- [PE10] Daniel P. Palomar and Yonina C. Eldar. *Convex optimization in signal processing and communications*. Cambridge university press, 2010.
- [Sun+20] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. “A Survey of Optimization Methods From a Machine Learning Perspective”. In: *IEEE Transactions on Cybernetics* 50.8 (2020), pp. 3668–3681. DOI: [10.1109/TCYB.2019.2950779](https://doi.org/10.1109/TCYB.2019.2950779).
- [Tay17] Adrien B. Taylor. “Convex Interpolation and Performance Estimation of first-order Methods for Convex Optimization”. PhD thesis. UCLouvain - Ecole polytechnique de Louvain, 2017. URL: https://dial.uclouvain.be/pr/boreal/en/object/boreal%5C%3A182881/datastream/PDF_01/view.
- [TD21] Adrien Taylor and Yoel Drori. “An optimal gradient method for smooth (possibly strongly) convex minimization”. 2021. URL: <https://arxiv.org/pdf/2101.09741.pdf>.

- [THG16] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. *Smooth Strongly Convex Interpolation and Exact Worst-case Performance of First-order Methods*. 2016. arXiv: [1502.05666 \[math.OC\]](https://arxiv.org/abs/1502.05666).
- [THG17a] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. “Performance estimation toolbox (PESTO): Automated worst-case analysis of first-order optimization methods”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 1278–1283. DOI: [10.1109/CDC.2017.8263832](https://doi.org/10.1109/CDC.2017.8263832).
- [THG17b] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. “Exact Worst-Case Performance of First-Order Methods for Composite Convex Optimization”. In: *SIAM Journal on Optimization* 27.3 (2017), pp. 1283–1313. ISSN: 1095-7189. DOI: [10.1137/16m108104x](https://doi.org/10.1137/16m108104x).
- [THG20a] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. *Exact worst-case convergence rates of the proximal gradient method for composite convex minimization*. 2020. arXiv: [1705.04398 \[math.OC\]](https://arxiv.org/abs/1705.04398).
- [THG20b] Adrien B. Taylor, Julien M. Hendrickx, and François Glineur. *Performance Estimation Toolbox (PESTO): User Manual*. 2020.
- [VFL18] Bryan Van Scoy, Randy A. Freeman, and Kevin M. Lynch. “The Fastest Known Globally Convergent First-Order Method for Minimizing Strongly Convex Functions”. In: *IEEE Control Systems Letters* 2.1 (2018), pp. 49–54. DOI: [10.1109/LCSYS.2017.2722406](https://doi.org/10.1109/LCSYS.2017.2722406).
- [WRJ18] Ashia C. Wilson, Benjamin Recht, and Michael I. Jordan. *A Lyapunov Analysis of Momentum Methods in Optimization*. 2018. arXiv: [1611.02635 \[math.OC\]](https://arxiv.org/abs/1611.02635).
- [Yan+19] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H. Johansson. “A survey of distributed optimization”. In: *Annual Reviews in Control* 47 (2019), pp. 278–305. DOI: <https://doi.org/10.1016/j.arcontrol.2019.05.006>.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl