Pesquisa Operacional (2014) 34(3): 395-419

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
doi: 10.1590/0101-7438.2014.034.03.0395

COMPLEXITY OF FIRST-ORDER METHODS FOR DIFFERENTIABLE CONVEX OPTIMIZATION

Clóvis C. Gonzaga[1] and Elizabeth W. Karas[2*]

Received December 8, 2013 / Accepted February 9, 2014

ABSTRACT. This is a short tutorial on complexity studies for differentiable convex optimization. A complexity study is made for a class of problems, an "oracle" that obtains information about the problem at a given point, and a stopping rule for algorithms. These three items compose a *scheme*, for which we study the performance of algorithms and problem complexity. Our problem classes will be quadratic minimization and convex minimization in $\mathbb{R}^n$. The oracle will always be first order. We study the performance of steepest descent and Krylov space methods for quadratic function minimization and Nesterov's approach to the minimization of differentiable convex functions.

Keywords: first-order methods, complexity analysis, differentiable convex optimization.

1 INTRODUCTION

Due to the huge increase in the size of problems tractable with modern computers, the study of problem complexity and algorithm performance became essential. This was recognized very early by computer scientists and mathematicians working on combinatorial problems, and has recently become a central issue in continuous optimization. Complexity studies for these problems started in the former Soviet Union, and the main results are described in the book by Nemirovski & Yudin [14].

The special case of Linear Programming, which will not be tackled in this paper, initiated with Khachiyan [10], also in Russia in 1978, and had an explosive expansion in the West with the creation of interior point methods in the 80's and 90's.

This paper starts with a brief introduction to the main concepts in the study of algorithm performance and complexity, following Nemirovski and Yudin, and then apply to the study of the convex optimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x), \tag{1}$$

*Corresponding author.
[1]Department of Mathematics, Federal University of Santa Catarina, Cx. Postal 5210, 88040-970 Florianópolis, SC, Brazil. E-mail: clovis@mtm.ufsc.br
[2]Department of Mathematics, Federal University of Paraná, Cx. Postal 19081, 81531-980 Curitiba, PR, Brazil. E-mail: ewkaras@ufpr.br; ewkaras@gmail.com

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function.

In Section 2 we introduce the general framework for the study of algorithm performance and problem complexity and present a simple example.

We dedicate Section 3 to study the special case of convex quadratic functions, because they are the simplest non-linear functions: if a method is inefficient for quadratic problems, it will certainly be inefficient for more general problems; if it is efficient, it has a good chance of being adaptable to general differentiable convex problems, because near an optimal solution the quadratic approximation of the function uses to be precise. We study the performance of steepest descent and of Krylov space methods.

Section 4 will describe and analyze a basic method for unconstrained convex optimization devised by Nesterov [15], with "accelerated steepest descent" iterations. This method has become very popular, and the presentation and complexity proofs will be based on our paper [7].

Finally, we comment in Section 5 on improvements of this basic algorithm, presenting without proofs its extension to problems restricted to "simple sets" (sets onto which projecting a vector is easy).

2 SCHEMES, PERFORMANCE AND COMPLEXITY

A complexity study is associated with a scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ as follows.

(i) Σ is a class of problems.
Examples: linear programming problems, unconstrained minimization of convex functions.

(ii) $\mathcal{O}$ is an oracle associated to Σ.
The oracle is responsible for accessing the available information $\mathcal{O}(x)$ about a given problem in Σ at a given point x.
Examples: $\mathcal{O}(x) = \{f(x)\}$ (zero order)

$$\mathcal{O}(x) = \{f(x), \nabla f(x)\} \quad \text{(first order)}.$$

(iii) τ_ε is a stopping rule, associated with a precision $\varepsilon > 0$.
Examples: for the minimization problem (1), τ_ε defined by
$f(x) - f^* \leq \varepsilon$,
$\|\nabla f(x)\| \leq \varepsilon$
$\|x - x^*\| \leq \varepsilon$
where x^* is a solution of the problem and $f^* = f(x^*)$.

An instance in a scheme would be for example: Solve a convex minimization problem (Σ) using only first order information $(\mathcal{O})$ to a precision $\|\nabla f(x^k)\| \leq 10^{-6}$ (τ_ε).

Algorithms. The general problem associated with a scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ is to find a point satisfying τ_ε, using as information only consultations to the oracle and any mathematical procedures that do not depend on the particular problem being solved.

The algorithms studied in this paper follow the *black box* model described now. An algorithm starts with a point x^0 and computes a sequence $(x^k)_{k\in\mathbb{N}}$. Each iteration k accesses the oracle at x^k and uses the information obtained by the oracle at $x^0, x^1, \ldots x^k$ to compute a new point x^{k+1}. It stops if x^{k+1} satisfies τ_ε.

Algorithm 1. *Black box model for* $(\Sigma, \mathcal{O}, \tau_\varepsilon)$

Data: $x^0, \varepsilon > 0, k = 0, I_{-1} = \emptyset$
WHILE x^k does not satisfy τ_ε
    Oracle at x^k: $\mathcal{O}(x^k)$
    Update information set: $I_k = I_{k-1} \cup \mathcal{O}(x^k)$
    Apply rules of the method to I_k: find x^{k+1}
    $k = k + 1$.

Algorithm performance for $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ (worst case performance)

Consider an algorithm for $(\Sigma, \mathcal{O}, \tau_\varepsilon)$.

- The *iteration performance* is a bound on the number of iterations (oracle calls) needed to solve any problem in the scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$. This bound will depend on ε and on parameters associated with each specific problem (initial point, space dimension, condition number, Lipschitz constants, etc.). In other words, it is the number of iterations needed to solve the "worst possible" problem in $(\Sigma, \mathcal{O}, \tau_\varepsilon)$.
- The *numerical performance* is a bound on the number of arithmetical operations needed in the worst case. The numerical performance is usually proportional to the iteration performance for each given algorithm. In this paper we only study the iteration performance of algorithms.
- The *complexity* of the scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ is the performance of the best possible algorithm for the scheme. It is frequently unknown, and finding it for different schemes is the main purpose of complexity studies.

The performance of any algorithm for a scheme gives an upper bound to its complexity. A lower bound to the complexity may sometimes be found by constructing an example of a (difficult) problem in Σ and finding a lower bound for any algorithm based on the same oracle and stopping rule. This will be the case in the end of Section 3.

If the performance of an algorithm for a scheme matches its complexity or a fixed multiple of it, it is called an *optimal algorithm* for the scheme.

First-order algorithms: In most of this paper we study first-order algorithms for solving the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function. The problem classes will be the special cases of quadratic and convex functions.

A first-order algorithm starts from a given point x^0 and constructs a sequence (x^k) using the oracle $\mathcal{O}(x^k) = \{f(x^k), \nabla f(x^k)\}$ or simply $\mathcal{O}(x^k) = \{\nabla f(x^k)\}$. Each step computes a point x^{k+1} using the information set $I_k = \cup_{j=0}^{k} \mathcal{O}(x^j)$ so that

$$x^{k+1} \in x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k)\right\}$$

where span(S) stands for the subspace generates by S.

In particular, the most well-known minimization algorithm is the steepest descent method, in which

$$x^{k+1} = x^k - \lambda_k \nabla f(x^k),$$

where λ_k is a steplength. Each different choice of steplength (the rules of the method) defines a different steepest descent algorithm. This will be studied ahead in this paper.

Remark: In our algorithm model we used a single oracle, but there may be more than one. Typically, $\mathcal{O}^0(x) = \{f(x)\}$, $\mathcal{O}^1(x) = \{\nabla f(x)\}$, and $\mathcal{O}^0$ may be called more than once in each iteration. This is the case when line searches are used. The performance evaluation must then be adapted.

The notation O(·). Given two real positive functions $g(\cdot)$ and $h(\cdot)$, we say that $g = \mathrm{O}(h)$ if there exists some constant $K > 0$ such that $g(\cdot) \leq Kh(\cdot)$. This notation is very useful in complexity studies. For example, we shall prove that a certain steepest descent algorithm stops for $k \leq \frac{C}{4}\log\left(\frac{1}{\varepsilon}\right)$, where C is a parameter that identifies the problem in Σ and ε is the precision. We may write $k = C\,\mathrm{O}(\log(1/\varepsilon))$, ignoring the coefficient $1/4$.

2.1 Example: root of a continuous function

Here we present a simple example to illustrate how a complexity analysis works. Consider the following example of $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ given by:

Σ: Given a continuous function $f : [0, 1] \to \mathbb{R}$ with $f(0) \leq 0$ and $f(1) \geq 0$, find $\bar{x} \in [0, 1]$ such that $f(\bar{x}) = 0$.

$\mathcal{O}$: For $x \in [0, 1]$, $\mathcal{O}(x) = \{f(x)\}$.

τ_ε: For $\varepsilon > 0$, τ_ε is satisfied if $|x - x^*| \leq \varepsilon$ for some root x^*.

Remark: The stopping rule above is obviously not computable. We shall use a practical rule that implies τ_ε.

Algorithm 2. *Bisection*

Data: $\varepsilon \in (0, 1)$, $a_0 = 0$, $b_0 = 1$, $k = 0$
WHILE $b_k - a_k > \varepsilon$ (stopping rule)
$m = (a_k + b_k)/2$
Compute $f(m)$ (oracle)
IF $f(m) \leq 0$, set $a_{k+1} = m$, $b_{k+1} = b_k$
ELSE set $b_{k+1} = m$, $a_{k+1} = a_k$
$k = k + 1$.

Performance: the following facts are straightforward at all iterations:

(i) $f(a_k) \leq 0$ and $f(b_k) \geq 0$ and by the intermediate value theorem, τ_ε is implied by $b_k - a_k \leq \varepsilon$.

(ii) $b_k - a_k = 2^{-k}$.

If the algorithm does not stop at iteration k, then $2^{-k} > \varepsilon$, and then $k < \log_2(1/\varepsilon)$. We conclude that the stopping rule will be satisfied for $k = \lceil \log_2(1/\varepsilon) \rceil$, where $\lceil r \rceil$ is the smallest integer above r. Thus that the performance of the scheme above is $k = \lceil \log_2(1/\varepsilon) \rceil = \mathrm{O}(\log(1/\varepsilon))$.

It is possible to prove that this is the best possible algorithm for this scheme, and hence the *complexity* of the scheme is this performance.

Remarks:

(i) Note that the only assumption here was the continuity of f. With stronger assumptions (Lipschitz constants for instance), better algorithms are described in numerical calculus textbooks.

(ii) The rules of the method are in the bisection calculation. Only the present oracle information $\mathcal{O}(m)$ is used at step k.

3 MINIMIZATION OF A QUADRATIC FUNCTION: FIRST-ORDER METHODS

Quadratic functions are the simplest nonlinear functions, and so an efficient algorithm for minimizing nonlinear functions must also be efficient in the quadratic case. On the other hand, near a minimizer, a twice differentiable function is usually well approximated by a quadratic function. A quadratic function is defined by

$$x \in \mathbb{R}^n \mapsto f(x) = c^T x + \frac{1}{2} x^T H x$$

where $c \in \mathbb{R}^n$ and H is an $n \times n$ symmetric matrix. Then for $x \in \mathbb{R}^n$,

$$\nabla f(x) = c + Hx, \quad \nabla^2 f(x) = H.$$

If x^* is a minimizer or a maximizer of f, then

$$\nabla f(x^*) = c + Hx^* = 0,$$

and hence finding an extremal of f is equivalent to solving the linear system $Hx^* = -c$, one of the most important problems in Mathematics.

The behavior of a quadratic function depends on the eigenvalues of its Hessian H. Since H is symmetric, it is known that H has n real eigenvalues

$$\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n,$$

which may be associated with n orthonormal (mutually orthogonal with unit norm) eigenvectors $v_1, v_2, \ldots, v_n$. There are four cases, represented in Figure 1:

(i) If $\mu_1 > 0$, then H is a positive definite matrix, f is strictly convex and its unique minimizer is the unique solution of $Hx = -c$.

(ii) If $\mu_1 < 0$, then $\inf_{x\in\mathbb{R}^n} f(x) = -\infty$, and $f(x) \to -\infty$ along the direction v_1.

Consider now the cases in which there are null eigenvalues. Let them be $\mu_1 = \mu_2 = \ldots = \mu_k = 0$. Thus H is a positive semi-definite matrix.

(iii) If $c^T v_i = 0$ for $i = 1, \ldots, k$, then f has a k-dimensional set of minimizers.

(iv) If $c^T v_i < 0$ for some $i = 1, \ldots, k$, then f is unbounded below.

In this section, we study the following scheme:

Σ: the class of quadratic functions that are bounded below (cases (i) and (iii) above). A function in Σ has at least one minimizer x^*. Without loss of generality, the study of algorithmic properties (not the implementation) may assume that $x^* = 0$, and so the function becomes

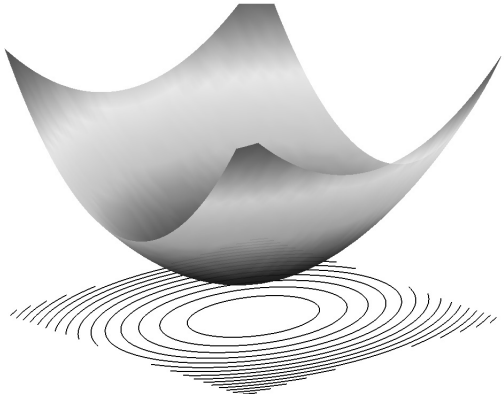
$$f(x) = \frac{1}{2}x^T Hx, \qquad \text{with} \qquad \nabla f(x) = Hx \qquad \text{and} \qquad f^* = f(x^*) = 0. \quad (2)$$

$\mathcal{O}$: $\mathcal{O}(x) = \{f(x), \nabla f(x)\}$ (first order).

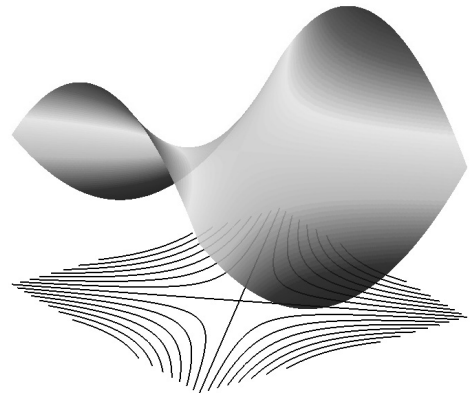
τ_ε: Given an initial point $x^0 \in \mathbb{R}^n$, two rules will be used in the analysis:
- Absolute error bound: $f(x) - f^* \leq \varepsilon$,
- Relative error bound: $f(x) - f^* \leq \varepsilon(f(x^0) - f^*)$.

These rules are not implementable, because they require the knowledge of x^*, but are very useful in the performance analysis.

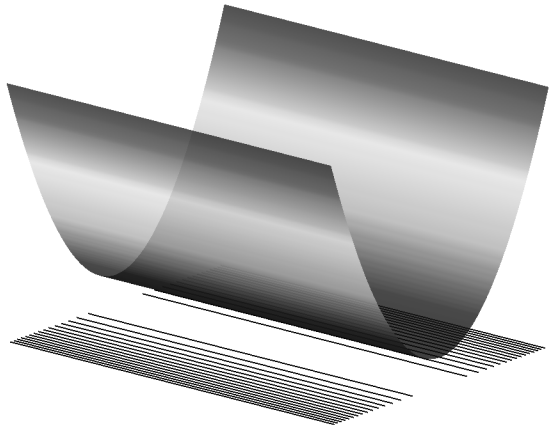
Simplification: As we explained above, we assume that $f(\cdot)$ has a minimizer $x^* = 0$. After performing the analysis with this simplification, we substitute $x - x^*$ for x. A further simplification may be done by diagonalizing H, also without loss of generality.



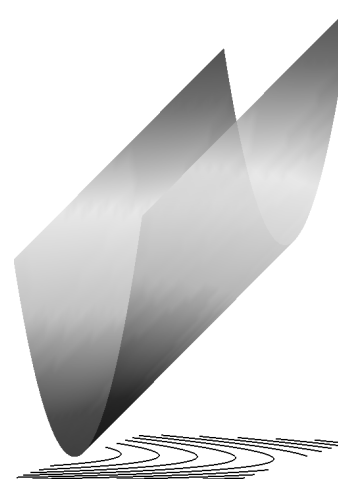
Case (i)



Case (ii)



Case (iii)



Case (iv)

Figure 1 – Quadratic functions.

3.1 Steepest descent algorithms

In the first half of the 19th century, Cauchy found that the gradient $\nabla f(x)$ of a function f is the direction of maximum ascent of f from x, and stated the gradient method. It is the most basic of all optimization algorithms, and its performance is still an active research topic.

Algorithm 3. *Steepest descent algorithm (model)*

Data: $x^0 \in \mathbb{R}^n$, $\varepsilon > 0$, $k = 0$
WHILE x^k does not satisfy τ_ε
    Choose a steplength $\lambda_k > 0$
    $x^{k+1} = x^k - \lambda_k \nabla f(x^k) = (I - \lambda_k H)x^k$
    $k = k + 1$.

Steplengths: Each different method for choosing the steplengths λ_k defines a different steepest descent algorithm. Let us describe the two best known choices for the steplengths:

- The Cauchy step, or exact step,

$$\lambda_k = \underset{\lambda \geq 0}{\operatorname{argmin}} f(x^k - \lambda \nabla f(x^k)), \tag{3}$$

  the unique minimizer of f along the direction $-g$ with $g = \nabla f(x^k)$. The steplength is computed by setting $\nabla f(x^k - \lambda_k g) \perp g$ and simplifying, which results in

$$\lambda_k = \frac{g^T g}{g^T H g}. \tag{4}$$

- The short step: $\lambda_k < 2/\mu_n$, a fixed steplength.

Complexity results

Now we study the iteration performance of the steepest descent methods with these two steplength choices for minimizing a strictly convex quadratic function (case (i)). Given $\varepsilon > 0$ and $x^0 \in \mathbb{R}^n$, we consider that τ_ε is satisfied at a given $x \in \mathbb{R}^n$ if

$$f(x) - f^* \leq \varepsilon \, (f(x^0) - f^*) \qquad \text{(relative error bound)}. \tag{5}$$

In both cases the algorithm stops in $O(C \log(1/\varepsilon))$ iterations, where $C = \mu_n/\mu_1$. At this moment the following question is open: find a steepest descent algorithm (by a different choice of λ_k) with performance $O(\sqrt{C}\log(1/\varepsilon))$. This performance is achieved in practice for "normal problems" (but not for particular worst case problems) by Barzilai-Borwein and spectral methods, described in [3].

Theorem 1. *Let $C = \mu_n/\mu_1 \geq 1$ be the condition number of H. The iteration performance of the steepest descent method with Cauchy steplength for minimizing f starting at $x^0 \in \mathbb{R}^n$ and with stopping criterion* (5) *is given by*

$$k \leq \left\lceil \frac{C}{4} \log\left(\frac{1}{\varepsilon}\right) \right\rceil.$$

Proof. We begin by stating a classical result for the steepest descent step, which is based on the Kantorovich inequality and is proved for instance in [12, p. 238],

$$f(x^k) \leq \left(\frac{C-1}{C+1}\right)^2 f(x^{k-1}).$$

Using this recursively, we obtain

$$\frac{f(x^k)}{f(x^0)} \leq \left(\frac{C-1}{C+1}\right)^{2k} = \left(\frac{C-1}{C+1}\right)^{\frac{2k}{C}C},$$

which implies

$$\log\left(\frac{f(x^k)}{f(x^0)}\right) \leq \frac{2k}{C}\log\left(\frac{C-1}{C+1}\right)^C.$$

It is known that $t \in [1, +\infty) \mapsto \left(\frac{t-1}{t+1}\right)^t$ is an increasing function and that for $t > 1$,

$$\left(\frac{t-1}{t+1}\right)^t \leq \lim_{t\to\infty}\left(\frac{t-1}{t+1}\right)^t = \frac{1}{e^2}.$$

Consequently

$$\log\left(\frac{f(x^k)}{f(x^0)}\right) \leq \frac{2k}{C}\log\left(\frac{1}{e^2}\right) = \frac{-4k}{C}.$$

If τ_ε is not satisfied at an iteration k, then by (5), $\frac{f(x^k)}{f(x^0)} > \varepsilon$ or

$$\log(\varepsilon) < \log\left(\frac{f(x^k)}{f(x^0)}\right) \leq -\frac{4k}{C},$$

which implies $k < \frac{C}{4}\log\left(\frac{1}{\varepsilon}\right)$, completing the proof. □

Example. In this example we show that the bound obtained in Theorem 1 is sharp, i.e., it cannot be improved. Take the following problem in $\mathbb{R}^2$:

$$f(x) = \frac{1}{2}x^T Hx \quad \text{with} \quad H = \text{diag}(1, C) \qquad (\text{i.e.} \quad \mu_1 = 1, \mu_2 = C).$$

Assume that the initial point of some iteration has the shape $x = (C, 1)\, z$, for some $z \in \mathbb{R}$. Then

$$\nabla f(x) = (x_1, Cx_2) = (1, 1)\, Cz.$$

Computing the steplength λ by (4), we obtain $\lambda = 2/(C+1)$, and then the next iterate will be

$$x^+ = (C, 1)\, z - \frac{2C}{C+1}\,(1, 1)\, z = \frac{C-1}{C+1}\,(C, -1)\, z.$$

It follows that

$$f(x^+) = \left(\frac{C-1}{C+1}\right)^2 f(x),$$

and this will be repeated on all iterations, with the worst possible performance as in Theorem 1.

Theorem 2. *Let $C = \mu_n/\mu_1 \geq 1$ be the condition number of H. The iteration performance of the steepest descent method with short steps $\lambda_k = 1/\mu_n$, for minimizing f starting at $x^0 \in \mathbb{R}^n$ and with stopping criterion* (5) *is given by*

$$k \leq \left\lceil \frac{C}{2}\log\left(\frac{1}{\varepsilon}\right)\right\rceil.$$

Proof. A simplification in the analysis can be made by diagonalizing the matrix H by using the orthonormal matrix whose columns are the eigenvectors of H. Thus, we can consider, without loss of generality, that $H = \text{diag}(\mu_1, \ldots, \mu_n)$. Given $x^0 \in \mathbb{R}^n$, by the steepest descent algorithm with short steps,

$$x^k = x^{k-1} - \frac{1}{\mu_n}\nabla f(x^{k-1}) = \left(I - \frac{1}{\mu_n}H\right)x^{k-1}.$$

Thus, for all $i = 1, \ldots, n$,

$$\left|x_i^k\right| = \left(1 - \frac{\mu_i}{\mu_n}\right)\left|x_i^{k-1}\right| \leq \left(1 - \frac{1}{C}\right)\left|x_i^{k-1}\right|.$$

Consequently, by the definition of f,

$$f(x^k) \leq \left(1 - \frac{1}{C}\right)^2 f(x^{k-1}).$$

Proceeding like in the proof of Theorem 1, we obtain

$$\log\left(\frac{f(x^k)}{f(x^0)}\right) \leq \frac{2k}{C}\log\left(\lim_{t\to\infty}\left(1 - \frac{1}{t}\right)^t\right) = \frac{2k}{C}\log\left(\frac{1}{e}\right) = \frac{-2k}{C}.$$

So, if τ_ε is not satisfied at an iteration k, then $k < \frac{C}{2}\log\left(\frac{1}{\varepsilon}\right)$, completing the proof. □

Remarks:

- When the short steps $1/\mu_n$ are used, the result is that for $i = 1, \ldots, n$, $|x_i^k| \leq \sqrt{\varepsilon}\,|x_i^0|$, for $k \geq \frac{C}{2}\log(1/\varepsilon)$. Hence not only $f(x^k) \leq \varepsilon f(x^0)$, but also $\|x^k\| \leq \sqrt{\varepsilon}\,\|x^0\|$ and $\|\nabla f(x^k)\| \leq \sqrt{\varepsilon}\,\|\nabla f(x^0)\|$.
- The diagonalization of H can be made without loss of generality for the performance analysis, as we did in the proof of Theorem 2. This leads to an interesting observation about the constant C, for the case in which there are null eigenvalues (case(iii)). Assuming that $\mu_1 = \mu_2 = \ldots = \mu_p = 0$, we see that for $i = 1, \ldots, p$, $(\nabla f(x))_i = 0$, and the variables x_i remain constant forever having no influence on the performance. The bounds in Theorems 1 and 2 remain valid for $C = \mu_n/\mu_{p+1}$.

3.2 Krylov methods

Krylov space methods are the best possible algorithms for minimizing a quadratic function using only first-order information. Let us describe the geometry of a Krylov space method for the quadratic (2).

- Starting at a point x^0, define the line $V_1 = \left\{x^0 + \theta\,\nabla f(x^0) \mid \theta \in \mathbb{R}\right\}$ and

$$x^1 = \underset{x\in V_1}{\text{argmin}}\, f(x) = x^0 + \theta\,\nabla f(x^0). \qquad (P_1)$$

  This is actually the Cauchy step. We may write $V_1 = x^0 + \text{span}\left\{Hx^0\right\}$.

- Second step: take the affine space defined by $\nabla f(x^0)$ and $\nabla f(x^1)$, $V_2 = x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1)\right\}$ and note that since $\nabla f(x^1) = H(x^0 + \theta\ \nabla f(x^0)) = Hx^0 + \theta\ H^2x^0$,
$$V_2 = x^0 + \text{span}\left\{Hx^0, H^2x^0\right\}$$
and the next iterate will be
$$x^2 = \underset{x \in V_2}{\text{argmin}}\ f(x). \qquad (P_2)$$
This is a two-dimensional problem.

- k-th step: adding $\nabla f(x^{k-1})$ to the set of gradients, we construct the set
$$V_k = x^0 + \text{span}\left\{Hx^0, \ldots, H^kx^0\right\}$$
and the next point will be
$$x^k = \underset{x \in V_k}{\text{argmin}}\ f(x), \qquad (P_k)$$
a $k-$dimensional problem.

Since $\nabla f(x^k) \perp V_k$ because of the minimization, either $\nabla f(x^k) = 0$ and the problem is solved, or V_{k+1} is $(k+1)-$dimensional. It is then clear that x^n is an optimal solution because $V_n = \mathbb{R}^n$. This gives us a first performance bound $k \leq n$ for the Krylov space method. This bound is bad for high-dimensional spaces.

Main question: how to solve (P_k). Without proof (see for instance [15, 20]), it is known that the directions $(x^k - x^{k-1})$ are conjugate, and any conjugate direction algorithm like Fletcher-Reeves [4, 18] solves (P_k) at each iteration with about the same work as in the steepest descent method.

From now on we do a performance analysis of the Krylov space method, with stopping criterion

$$f(x^k) - f^* \leq \varepsilon \qquad \text{(absolute error bound)}.$$

A result for the relative error bound will also be discussed in the end of the section. The analysis is quite technical and will result in (16).

Definition 1. *Given $x^0 \in \mathbb{R}^n$ and $k \in \mathbb{N}$, define the k-th Krylov space by*

$$\mathcal{K}_k = \text{span}\left\{Hx^0, H^2x^0, \ldots, H^kx^0\right\}.$$

Consider $V_k = x^0 + \mathcal{K}_k$ and define the sequence (x^k) by

$$x^k = \underset{x \in V_k}{\text{argmin}}\ f(x). \qquad (6)$$

Let $\mathcal{P}_k$ be the set of polynomials $p : \mathbb{R} \to \mathbb{R}$ of degree k such that $p(0) = 1$, i.e,

$$\mathcal{P}_k = \left\{1 + a_1t + a_2t^2 + \cdots + a_kt^k \mid a_i \in \mathbb{R}, i = 1, \ldots, k\right\}. \qquad (7)$$

From now on we deal with matrix polynomials, setting $t = H$.

Lemma 1. *A point $x \in V_k$ if, and only if, $x = p(H)x^0$ for some polynomial $p \in \mathcal{P}_k$. Furthermore,*

$$f(x) = \frac{1}{2}(x^0)^T H\big(p(H)\big)^2 x^0. \tag{8}$$

Proof. A point $x \in V_k$ if, and only if,

$$x = x^0 + a_1 Hx^0 + a_2 H^2 x^0 + \cdots + a_k H^k x^0 = p(H)x^0,$$

where $p \in \mathcal{P}_k$. Furthermore,

$$f(x) = \frac{1}{2}(x^0)^T \big(p(H)\big)^T H p(H) x^0.$$

As H is symmetric, $\big(p(H)\big)^T H = Hp(H)$, completing the proof. □

Lemma 2. *For any polynomial $p \in \mathcal{P}_k$,*

$$f(x^k) \leq \frac{1}{2}(x^0)^T H\big(p(H)\big)^2 x^0.$$

Proof. Consider an arbitrary polynomial $p \in \mathcal{P}_k$. From Lemma 1, the point $x = p(H)x^0$ belongs to V_k. As x^k minimizes f in V_k, we have $f(x^k) \leq f(x)$. Using (8) we complete the proof. □

Lemma 3. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. If $q : \mathbb{R} \to \mathbb{R}$ is a polynomial, then $q(\lambda_1), q(\lambda_2), \ldots, q(\lambda_n)$ are the eigenvalues of $q(A)$.*

Proof. As A is a symmetric matrix, there exists an orthogonal matrix P such that $A = PDP^T$, with $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. If $q(t) = a_0 + a_1 t + \cdots + a_k t^k$, then

$$q(H) = a_0 I + a_1 PDP^T + \cdots + a_k (PDP^T)^k = P\left(a_0 I + a_1 D + \cdots + a_k D^k\right) P^T.$$

Note that

$$a_0 I + a_1 D + \cdots + a_k D^k = \operatorname{diag}\big(q(\lambda_1), q(\lambda_2), \ldots, q(\lambda_n)\big)$$

which completes the proof. □

3.2.1 Chebyshev Polynomials

The Chebyshev polynomials will be needed in the performance analysis of Krylov methods.

Definition 2. *The Chebyshev polynomial of degree k, $T_k : [-1, 1] \to \mathbb{R}$, is defined by*

$$T_k(t) = \cos\big(k \arccos(t)\big).$$

The next lemma shows that T_k is, in fact, a polynomial (even though it does not look like one).

Lemma 4. *For all $t \in [-1, 1]$, $T_0(t) = 1$ and $T_1(t) = t$. Furthermore, for all $k \geq 1$,*

$$T_{k+1}(t) = 2t\,T_k(t) - T_{k-1}(t).$$

Proof. The first statements follow from the definition. In order to prove the recurrence rule, consider $\theta : [-1, 1] \to [0, \pi]$, given by $\theta(t) = \arccos(t)$. Thus,

$$T_{k+1}(t) = \cos\big((k+1)\theta(t)\big) = \cos\big(k\theta(t)\big)\cos\big(\theta(t)\big) - \sin\big(k\theta(t)\big)\sin\big(\theta(t)\big)$$

and

$$T_{k-1}(t) = \cos\big((k-1)\theta(t)\big) = \cos\big(k\theta(t)\big)\cos\big(\theta(t)\big) + \sin\big(k\theta(t)\big)\sin\big(\theta(t)\big).$$

But $\cos\big(k\theta(t)\big) = T_k(t)$ and $\cos\big(\theta(t)\big) = t$. So,

$$T_{k+1}(t) + T_{k-1}(t) = 2t\,T_k(t),$$

completing the proof. □

Lemma 5. *If $T_k(t) = a_k t^k + \cdots + a_2 t^2 + a_1 t + a_0$, then $a_k = 2^{k-1}$. Furthermore,*

(*i*) *If k is even, then $a_0 = (-1)^{\frac{k}{2}}$ and $a_{2j-1} = 0$, for all $j = 1, \ldots, \frac{k}{2}$;*

(*ii*) *If k is odd, then $a_1 = (-1)^{\frac{k-1}{2}} k$ and $a_{2j} = 0$, for all $j = 0, 1, \ldots, \frac{k-1}{2}$.*

Proof. We prove by induction. The results are trivial for $k = 0$ and $k = 1$. Suppose that the results hold for all natural number less than or equal to k. Using the induction hypothesis, consider

$$T_k(t) = 2^{k-1}t^k + \cdots + a_1 t + a_0 \quad \text{and} \quad T_{k-1}(t) = 2^{k-2}t^{k-1} + \cdots + b_1 t + b_0.$$

By Lemma 4,

$$T_{k+1}(t) = 2t\,(2^{k-1}t^k + \cdots + a_1 t + a_0) - (2^{k-2}t^{k-1} + \cdots + b_1 t + b_0), \tag{9}$$

leading to the first statement. Suppose that $(k+1)$ is even. Then k is odd and $(k-1)$ is even. Thus, by induction hypothesis, T_k has only odd powers of t and T_{k-1} has only even powers. In this way, by (9), T_{k+1} has only even powers of t. Furthermore, its independent term is

$$-b_0 = -(-1)^{\frac{k-1}{2}} = (-1)^{\frac{k+1}{2}}.$$

On the other hand, if $(k+1)$ is odd, then k is even and $(k-1)$ is odd. Again by the induction hypothesis, T_k only has even powers of t and T_{k-1} has only odd powers. Thus by (9), T_{k+1} has only odd powers of t. Furthermore, its linear term is

$$2ta_0 - b_1 t = 2t(-1)^{\frac{k}{2}} - (-1)^{\frac{k-2}{2}}(k-1)t = (-1)^{\frac{k}{2}}(k+1)t,$$

completing the proof. □

The next lemma discusses a relationship between a Chebyshev polynomial of odd degree and polynomials of the set $\mathcal{P}_k$, defined in (7).

Lemma 6. *Consider $L > 0$ and $k \in \mathbb{N}$. Then there exists $p \in \mathcal{P}_k$ such that, for all $t \in [0, L]$,*

$$T_{2k+1}\left(\frac{\sqrt{t}}{\sqrt{L}}\right) = (-1)^k(2k+1)\frac{\sqrt{t}}{\sqrt{L}}p(t).$$

Proof. By Lemma 5, for all $t \in [-1, 1]$, we have

$$T_{2k+1}(t) = t\left(2^{2k}t^{2k} + \cdots + (-1)^k(2k+1)\right),$$

where the polynomial in parentheses has only even powers of t. So, for all $t \in [0, L]$,

$$T_{2k+1}\left(\frac{\sqrt{t}}{\sqrt{L}}\right) = \frac{\sqrt{t}}{\sqrt{L}}\left(2^{2k}\left(\frac{t}{L}\right)^k + \cdots + (-1)^k(2k+1)\right).$$

Defining

$$p(t) = \frac{1}{(-1)^k(2k+1)}\left(2^{2k}\left(\frac{t}{L}\right)^k + \cdots + (-1)^k(2k+1)\right),$$

we complete the proof. □

3.2.2 Complexity results

Now we present the main result about the performance of Krylov methods for minimizing a convex quadratic function. This result is based on [19, Thm. 3, p. 170].

We use the matrix norm defined by

$$\|A\| = \sup\{\|Ax\| \mid \|x\| = 1\} = \max\{|\lambda| \mid \lambda \text{ is an eigenvalue of } A\}. \tag{10}$$

Theorem 3. *Let μ_n be the largest eigenvalue of H and consider the sequence (x^k) defined by* (6). *Then for $k \in \mathbb{N}$*

$$f(x^k) - f^* \leq \frac{\mu_n \|x^0 - x^*\|^2}{2(2k+1)^2}, \tag{11}$$

and $f(x^k) - f^* \leq \varepsilon$ is satisfied for

$$k \leq \left\lceil \frac{1}{\sqrt{8}} \frac{\sqrt{\mu_n}\|x^0 - x^*\|}{\sqrt{\varepsilon}} \right\rceil. \tag{12}$$

Proof. Without loss of generality, assume that $x^* = 0$. By Lemma 2, for all polynomial $p \in \mathcal{P}_k$,

$$f(x^k) \leq \frac{1}{2}(x^0)^T H\big(p(H)\big)^2 x^0 \leq \frac{1}{2}\|x^0\|^2 \left\| H\big(p(H)\big)^2 \right\|. \tag{13}$$

But, from Lemma 3 and (10),

$$\left\| H\big(p(H)\big)^2 \right\| = \max\left\{ \mu_i \big(p(\mu_i)\big)^2 \mid \mu_i \text{ is an eigenvalue of } H \right\}.$$

Considering the polynomial $p \in \mathcal{P}_k$ given in Lemma 6 and using the fact that all eigenvalues of H belong to $(0, \mu_n]$, we have

$$\begin{aligned} \left\| H\big(p(H)\big)^2 \right\| &\leq \max_{t\in[0,\mu_n]} \left\{ t\big(p(t)\big)^2 \right\} = \frac{\mu_n}{(2k+1)^2} \max_{t\in[0,\mu_n]} \left\{ T_{2k+1}^2\left(\frac{\sqrt{t}}{\sqrt{\mu_n}}\right)\right\} \\ &\leq \frac{\mu_n}{(2k+1)^2}, \end{aligned} \tag{14}$$

proving (11). If τ_ε is not satisfied at an iteration k, then $f(x^k) > \varepsilon$ and consequently

$$\varepsilon < \frac{\mu_n \|x^0\|^2}{2(2k+1)^2} < \frac{\mu_n \|x^0\|^2}{8k^2}$$

which implies (12) and completes the proof. □

Performance of the method for the relative error bound: A similar analysis for τ_ε given by (5), also using Chebyshev polynomials, can be done using the condition number C. This is done in [14, 23], and the result is

$$k \leq \frac{\sqrt{C}}{2} \log\left(\frac{2}{\varepsilon}\right) = \mathrm{O}\left(\sqrt{C}\, \log\left(\frac{1}{\varepsilon}\right)\right),$$

clearly better than the best performance of the steepest descent algorithm for the steplength rules studied above, and for reasonable values of μ_1, better than (12).

Complexity bound. The Krylov space methods uses at each iteration all the information gathered in the previous steps, and hence it seems to be the best possible algorithm based on first order information. In fact, Nemirovskii & Yudin [14] prove that no algorithm using a first order oracle can have a performance more than twice as good as the Krylov space method.

For methods based on accumulated first order information there is a negative result described by Nesterov [15, p. 59]: he constructs a quadratic problem (which he calls "the worst problem in the world") for which such methods need at least

$$k = \frac{3}{32} \frac{\sqrt{\mu_n}\, \|x^0 - x^*\|}{\sqrt{\varepsilon}} \tag{15}$$

iterations to reach τ_ε.

We conclude that the best performance for a first order method must be between the bounds (12) and (15). So the complexity of the scheme is

$$k = \sqrt{\mu_n}\, \|x^0 - x^*\|\, \mathrm{O}\left(\frac{1}{\sqrt{\varepsilon}}\right). \tag{16}$$

4 CONVEX DIFFERENTIABLE FUNCTIONS: THE BASIC ALGORITHM

In this section we study the performance of algorithms for the unconstrained minimization of differentiable convex functions. Quadratic functions are a particular case, and hence the performance bounds for first order algorithms will not be better than those found in the former section.

The role played by μ_n in quadratic functions will be played by a Lipschitz constant L for the gradient of f (indeed, for a quadratic function the largest eigenvalue is a Lipschitz constant for the gradient), and we shall see that there are optimal algorithms, i.e., algorithms with the performance given by (16) with μ_n replaced by L. These algorithms were developed by Nesterov [15], and are also studied in our papers [7, 8].

Consider the scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ where

Σ: the class of minimization problems of a convex continuously differentiable function f, with a Lipschitz constant $L > 0$ for the gradient. It means that for all $x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|. \tag{17}$$

$\mathcal{O}$: $\mathcal{O}(x) = \{f(x), \nabla f(x)\}$ (first order)

τ_ε: defined by $f(x) - f^* \leq \varepsilon$ where x^* is a solution of the problem and $f^* = f(x^*)$.

Simple quadratic functions. The following definition will be useful in our development: we shall call "simple" a quadratic function $\phi : \mathbb{R}^n \to \mathbb{R}$ with $\nabla^2\phi(x) = \gamma I$, $\gamma \in \mathbb{R}$, $\gamma > 0$. The following facts are easily proved for such functions:

- $\phi(\cdot)$ has a unique minimizer $v \in \mathbb{R}^n$ (which we shall refer as the *center* of the quadratic), and the function can be written as

$$x \in \mathbb{R}^n \mapsto \phi(v) + \frac{\gamma}{2}\|x - v\|^2. \tag{18}$$

- Given $x \in \mathbb{R}^n$,

$$v = x - \frac{1}{\gamma}\nabla\phi(x), \tag{19}$$

and

$$\phi(x) - \phi(v) = \frac{1}{2\gamma}\|\nabla\phi(x)\|^2. \tag{20}$$

4.1 The algorithm

We now state the main algorithm and then study its properties. We include in the statement of the algorithm the definitions of the relevant functions (approximations of $f(\cdot)$ and the simple quadratic defined below).

We begin by summarizing the geometrical construction at an iteration k, represented in Figure 2. The iteration starts with two points x^k, $v^k \in \mathbb{R}^n$ and a simple quadratic function

$$\phi_k(x) = f(x^k) + \frac{\gamma_k}{2}\|x - v^k\|^2,$$

whose global minimizer is v^k.

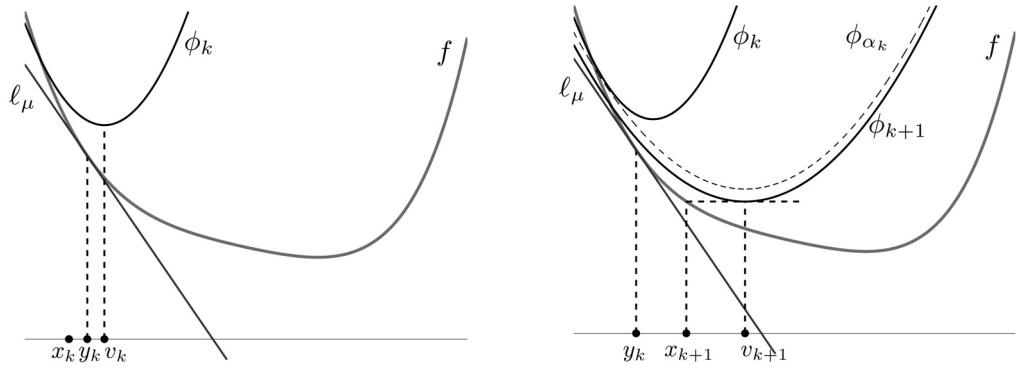


Figure 2 – The mechanics of the algorithm.

A point $y^k = x^k + \alpha(v^k - x^k)$ is chosen between x^k and v^k. The choice of α is a central issue, and will be discussed later. All the action is centered on y^k, with the following construction:

- Take a gradient step from y^k, generating x^{k+1}.
- Define a linear approximation of $f(\cdot)$

$$x \in \mathbb{R}^n \mapsto \ell(x) = f(y_k) + \nabla f(y_k)^T(x - y_k).$$

- Compute a value $\alpha \in (0, 1)$, and define $\phi_\alpha(x) = \alpha\ell(x) + (1 - \alpha)\phi_k(x)$, with Hessian $\gamma_{k+1}I = \nabla^2\phi_\alpha(x) = (1 - \alpha)\gamma_k I$, and let v^{k+1} be the minimizer of this simple quadratic. The iteration is completed by defining

$$\phi_{k+1}(x) = f(x^{k+1}) + \frac{\gamma_{k+1}}{2}\|x - v^{k+1}\|^2.$$

Now we state the algorithm.

Algorithm 4.

Data: $x_0 \in \mathbb{R}^n$, $v_0 = x_0$, $\gamma_0 = L$, $k = 0$

REPEAT

Compute $\alpha_k \in (0, 1)$ such that $L\alpha_k^2 = (1 - \alpha_k)\gamma_k$

Set $y^k = x^k + \alpha_k(v^k - x^k)$
Compute $f(y^k)$ and $g = \nabla f(y^k)$
Updates
$x^{k+1} = y^k - g/L$ (steepest descent step)
$\gamma_{k+1} = (1 - \alpha_k)\gamma_k$
For the analysis define
$x \mapsto \phi_k(x) = f(x^k) + \frac{\gamma_k}{2}\|x - v^k\|^2$
$x \mapsto \ell(x) = f(y^k) + g^T(x - y^k)$
$x \mapsto u(x) = f(y^k) + g^T(x - y^k) + \frac{L}{2}\|x - y^k\|^2$
$x \mapsto \phi_{\alpha_k}(x) = \alpha_k \ell(x) + (1 - \alpha_k)\phi_k(x)$
$v^{k+1} = \operatorname{argmin} \phi_{\alpha_k}(\cdot) = v^k - \dfrac{\alpha_k}{\gamma_{k+1}} g$
$k = k + 1$.

4.1.1 Analysis of the algorithm

The most important procedure in the algorithm is the choice of the parameter α_k, which then determines y^k at each iteration. The choice of α_k is the one devised by Nesterov in [15, Scheme (2.2.6)]. Instead of "discovering" the values for these parameters, we shall simply adopt them and show their properties.

Once y^k is determined, two independent actions are taken:

(i) A steepest descent step from y^k computes x^{k+1}.

(ii) A new simple quadratic is constructed by combining $\phi_k(\cdot)$ and the linear approximation $\ell(\cdot)$ of $f(\cdot)$ about y^k:

$$\phi_{\alpha_k}(x) = \alpha_k \ell(x) + (1 - \alpha_k)\phi_k(x).$$

Our scope will be to prove two facts:

- At any iteration k, $\phi^*_{\alpha_k} \geq f(x^{k+1})$.
- For all $x \in \mathbb{R}^n$, $\phi_{k+1}(x) - f(x) \leq (1 - \alpha_k)(\phi_k(x) - f(x))$.

From these facts we shall conclude that $f(x^k) \to f^*$ with the same speed as $\gamma_k \to 0$, which easily leads to the desired performance result.

The first lemma shows our main finding about the geometry of these points. All the action happens in the two-dimensional space defined by x^k, v^k, v^{k+1}. Note the beautiful similarity of the triangles in Figure 3.

Lemma 7. *Consider the sequences generated by Algorithm 4. Then for $k \in \mathbb{N}$,*

$$x^{k+1} - x^k = \alpha_k(v^{k+1} - x^k).$$

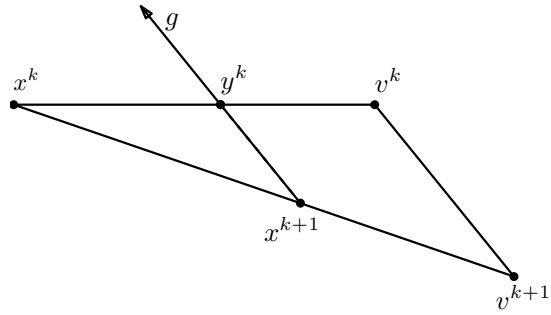


Figure 3 – Geometric properties of the steps.

Proof. By the algorithm, we know that $L\alpha_k^2 = \gamma_{k+1}$, and

$$\begin{aligned}
\alpha_k(v^{k+1} - x^k) &= \alpha_k\left(v^k - x^k - \frac{\alpha_k}{\gamma_{k+1}}g\right) \\
&= \alpha_k(v^k - x^k) - \frac{\alpha_k^2}{\gamma_{k+1}}g \\
&= \alpha_k(v^k - x^k) - \frac{1}{L}g \\
&= y^k - x^k - \frac{1}{L}g \\
&= x^{k+1} - x^k,
\end{aligned}$$

completing the proof. □

Lemma 8. *Consider the sequences generated by Algorithm 4. Then for $k \in \mathbb{N}$,*

$$f(y^k) \leq \phi_{\alpha_k}(v^k).$$

Proof. By the definition of ϕ_{α_k},

$$\phi_{\alpha_k}(v^k) = \alpha_k\ell(v^k) + (1-\alpha_k)\phi_k(v^k).$$

But, $\phi_k(v^k) = f(x^k) \geq \ell(x^k)$. Using this, the definition of ℓ and the fact that $\alpha_k \in (0,1)$, we have

$$\begin{aligned}
\phi_{\alpha_k}(v^k) &\geq \alpha_k\ell(v^k) + (1-\alpha_k)\ell(x^k) \\
&= \alpha_k\left(f(y^k) + g^T(v^k - y^k)\right) + (1-\alpha_k)\left(f(y^k) + g^T(x^k - y^k)\right) \\
&= f(y^k) + g^T\left(\alpha_k(v^k - y^k) + (1-\alpha_k)(x^k - y^k)\right). \qquad (21)
\end{aligned}$$

By the definition of y^k in the algorithm, $v^k - y^k = (1-\alpha_k)(v^k - x^k)$ and $x^k - y^k = -\alpha_k(v^k - x^k)$. Substituting this in (21), we complete the proof. □

Lemma 9. *Consider the sequences generated by Algorithm 4. Then for $k \in \mathbb{N}$,*

$$f(x^{k+1}) \leq u(x^{k+1}) \leq \phi_{\alpha_k}(v^{k+1}) = \phi_{\alpha_k}^*, \tag{22}$$

$$\phi_{k+1}(\cdot) \leq \phi_{\alpha_k}(\cdot). \tag{23}$$

Proof. The first inequality follows trivially from the convexity of f and the definition of u.

Since x^{k+1} and v^{k+1} are respectively global minimizers of $u(\cdot)$ and $\phi_{\alpha_k}(\cdot)$, we have from (18) that, for all $x \in \mathbb{R}^n$,

$$u(x) = u(x^{k+1}) + \frac{L}{2}\|x - x^{k+1}\|^2 \qquad \text{and} \qquad \phi_{\alpha_k}(x) = \phi_{\alpha_k}^* + \frac{\gamma_{k+1}}{2}\|x - v^{k+1}\|^2. \tag{24}$$

As $f(y^k) = u(y^k)$ and, from the last lemma, $u(y^k) \leq \phi_{\alpha_k}(v^k)$, we only need to show that

$$u(y^k) - u(x^{k+1}) = \phi_{\alpha_k}(v^k) - \phi_{\alpha_k}^*.$$

The construction is shown in Fig. 3: since, by Lemma 7, $x^{k+1} = x^k + \alpha_k(v^{k+1} - x^k)$,

$$y^k - x^{k+1} = \alpha_k(v^k - v^{k+1}).$$

Using this, (24) and the fact that by construction, $\alpha_k^2 = \frac{\gamma_{k+1}}{L}$, we obtain

$$u(y^k) - u(x^{k+1}) = \frac{L\alpha_k^2}{2}\|v^k - v^{k+1}\|^2 = \frac{\gamma_{k+1}}{2}\|v^k - v^{k+1}\|^2 = \phi_{\alpha_k}(v^k) - \phi_{\alpha_k}^*,$$

proving the second inequality of (22).

By construction,

$$\phi_{k+1}(x) = f(x^{k+1}) + \frac{\gamma_{k+1}}{2}\|x - v^{k+1}\|^2.$$

Comparing to (24) and using the fact that $f(x^{k+1}) \leq \phi_{\alpha_k}^*$, we get (23), completing the proof. □

Lemma 10. *For any $x \in \mathbb{R}^n$ and $k \in \mathbb{N}$,*

$$\phi_k(x) - f(x) \leq \frac{\gamma_k}{\gamma_0}\left(\phi_0(x) - f(x)\right). \tag{25}$$

Proof. By the definition of ϕ_{α_k} and the fact that $\ell(x) \leq f(x)$, for all $x \in \mathbb{R}^n$,

$$\phi_{\alpha_k}(x) \leq \alpha_k f(x) + (1 - \alpha_k)\phi_k(x).$$

Subtracting $f(x)$ in both sides, using (23) and the definition of γ_{k+1}, we have

$$\begin{aligned}\phi_{k+1}(x) - f(x) &\leq \phi_{\alpha_k}(x) - f(x)\\ &\leq (1 - \alpha_k)(\phi_k(x) - f(x))\\ &= \frac{\gamma_{k+1}}{\gamma_k}(\phi_k(x) - f(x)).\end{aligned}$$

Using this recursively, we get the result and complete the proof. □

4.1.2 Complexity

The following lemma was proved by Nesterov [15, p. 77] with a different notation.

Lemma 11. *Consider the sequence (γ_k) generated by Algorithm 4, i.e., given $\gamma_0 > 0$,*

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k, \qquad L\alpha_k^2 = \gamma_{k+1}.$$

Then, for $k \in \mathbb{N}$, $\gamma_k \leq 4L/k^2$.

Proof. As $\alpha_k = \sqrt{\gamma_{k+1}/L}$,

$$\gamma_{k+1} = \left(1 - \frac{1}{\sqrt{L}}\sqrt{\gamma_{k+1}}\right)\gamma_k.$$

Thus, the result follows directly from [7, Lemma 10]. □

Theorem 4. *Consider the sequences generated by Algorithm 4 and assume that x^* is an optimal solution. Then for $k \in \mathbb{N}$,*

$$f(x^k) - f^* \leq \frac{4L}{k^2}\|x^* - x_0\|^2, \tag{26}$$

and $f(x^k) - f^* \leq \varepsilon$ is satisfied for

$$k \leq \left\lceil 2\,\frac{\sqrt{L}\,\|x^0 - x^*\|}{\sqrt{\varepsilon}} \right\rceil. \tag{27}$$

Proof. From Lemma 10, (25) holds in particular at x^*,

$$\phi_k(x^*) - f^* \leq \frac{\gamma_k}{\gamma_0}\left(\phi_0(x^*) - f^*\right).$$

Using the fact that $f(x^k) = \phi_k(v^k) \leq \phi_k(x^*)$ and the definition of ϕ_0, we get,

$$f(x^k) - f^* \leq \frac{\gamma_k}{\gamma_0}\left(f(x_0) + \frac{\gamma_0}{2}\|x^* - x_0\|^2 - f^*\right). \tag{28}$$

Since x^* is a minimizer of the convex function f,

$$f(x^0) - f^* \leq \frac{L}{2}\|x^* - x^0\|^2.$$

Applying this and the result of Lemma 11 in (28),

$$f(x^k) - f^* \leq \frac{2L(L + \gamma_0)}{\gamma_0 k^2}\|x^* - x_0\|^2.$$

As $\gamma_0 = L$, we have (26). If τ_ε is not satisfied at an iteration k, then $f(x^k) - f^* > \varepsilon$ and consequently

$$\varepsilon < \frac{4L}{k^2}\|x^* - x_0\|^2,$$

which implies (27) and completes the proof. □

So, the iteration performance of Algorithm 4 is

$$k = \sqrt{L}\,\|x^0 - x^*\|\,\mathrm{O}\left(1/\sqrt{\varepsilon}\right),$$

which corresponds to the complexity (16) for quadratic performance. Then, the algorithm is optimal.

5 CONVEX DIFFERENTIABLE FUNCTIONS: ENHANCED ALGORITHMS

The algorithm presented in the former section may be extended in several ways: the need for a previous knowledge of a Lipschitz constant L may be eliminated, a strong convexity constant akin to the smallest eigenvalue in the quadratic case may be used, and the algorithm may be extended to problems constrained to so-called simple sets. These extensions are treated in our paper [8] and in references therein.

In this section we state the extension of the basic algorithm to problems with simple constraints, without proofs. Consider the scheme $(\Sigma, \mathcal{O}, \tau_\varepsilon)$ where

Σ: the class of problems

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega, \end{array}$$

where $\Omega \subset \mathbb{R}^n$ is a closed convex set and $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable, with a Lipschitz constant $L > 0$ for the gradient. We assume that Ω is a "simple" set, in the following sense: given an arbitrary point $x \in \mathbb{R}^n$, an oracle is available to compute $P_\Omega(x) = \operatorname{argmin}_{y\in\Omega} \|x - y\|$, the orthogonal projection onto the set Ω.

$\mathcal{O}$: $\mathcal{O}(x) = \{f(x), \nabla f(x), P_\Omega(x)\}$.

τ_ε: defined by $f(x) - f^* \leq \varepsilon$ where x^* is a solution of the problem and $f^* = f(x^*)$.

We now state the basic algorithm for constrained problems, without proofs. We keep in the statement the definition of the functions used in the analysis made in [8].

Algorithm 5.

Data: $x_0 \in \Omega$, $v_0 = x_0$, $\gamma_0 = L$, $k = 0$
REPEAT
  Compute $\alpha \in (0, 1)$ such that $L\alpha^2 = (1 - \alpha)\gamma_k$
  $y^k = x^k + \alpha(v^k - x^k)$
  Compute $f(y^k)$ and $g = \nabla f(y^k)$
  Updates
    $x^{k+1} = P_\Omega(y^k - g/L)$ (projected steepest descent step)
    $\gamma_{k+1} = (1 - \alpha_k)\gamma_k$

For the analysis define

$$x \mapsto \phi_k(x) = f(x^k) + \tfrac{\gamma_k}{2}\|x - v^k\|^2$$
$$x \mapsto \ell(x) = f(y^k) + g^T(x - y^k)$$
$$x \mapsto u(x) = f(y^k) + g^T(x - y^k) + \tfrac{L}{2}\|x - y^k\|^2$$
$$x \mapsto \phi_{\alpha_k}(x) = \alpha_k \ell(x) + (1 - \alpha_k)\phi_k(x)$$
$$v^{k+1} = \operatorname*{argmin}_{x\in\Omega} \phi_{\alpha_k}(x) = P_\Omega\left(v^k - \frac{\alpha_k}{\gamma_{k+1}}\, g\right)$$
$$k = k + 1.$$

Consider the sequences generated by Algorithm 5. Then, as proved in [8, Thm. 2.6], at any iteration k before stopping,

$$\varepsilon \le f(x^k) - f^* \le \frac{4}{k^2}\left(f(x_0) - f^* + \frac{L}{2}\,\|x^* - x_0\|^2\right).$$

and hence

$$k \le \frac{2}{\sqrt{\varepsilon}}\left(f(x_0) - f^* + \frac{L}{2}\,\|x^* - x_0\|^2\right)^{1/2}.$$

This expression is similar to (27). In fact, if x^* is a global minimizer, then

$$f(x^0) - f^* \le \frac{L}{2}\,\|x^* - x_0\|^2$$

may be introduced in the last expression, retrieving (27).

Estimations of the Lipschitz constant. Both Algorithms 4 and 5 and the algorithms discussed by Nesterov in [15, Chapter 2] make explicit use of a Lipschitz constant L for the function gradient. In [16], Nesterov describes a method for a more general problem, easily applied to the situations studied in this paper. This method includes a scheme for estimating the Lipschitz constant. In [7, 8], the authors eliminate the use of L at the cost of an extra imprecise line search, and obtain an algorithm which keeps the optimal complexity properties and also inherits the global convergence properties of the steepest descent method for general continuously differentiable optimization. Besides this, the algorithm takes advantage of the knowledge of the strong convexity constant for the function and develop in [7] an adaptive procedure for estimating it. In another context – constrained minimization of non-smooth homogeneous convex functions – Richtárik [21] uses an adaptive scheme for guessing bounds for the distance between a point x_0 and an optimal solution x^*. This bound determines the number of subgradient steps needed to obtain a desired precision.

Extensions. Nesterov's approach is applied to penalty methods by Lan, Lu & Monteiro [11], and interior descent methods based on Bregman distances are described by Auslender & Teboulle [1]. This method has been generalized to a non-interior method using projections by Rossetto [22]. Results for higher order methods are discussed in Nesterov & Polyak [17]. Accelerated versions of first-order algorithms following Nesterov's approach were developed by Monteiro, Ortiz & Svaiter [13] and by Beck & Teboulle [2], with improved performance for benchmark problems.

6 CONCLUSIONS

In this paper we described what we believe to be the basic results in the study of algorithm performance and problem complexity for the minimization of convex functions, both unconstrained and with simple constraints.

Algorithms with proved low worst-case performance are not necessarily efficient for practical problems. Khachiyan's algorithm [10] for linear programming is very inefficient, but had a great impact on the development of both continuous and discrete optimization. Karmarkar's algorithm [9] for linear programming improved Khachiyan's performance bound, and his bound was again improved later (see [6]). The effort to improve complexity led to better algorithms, which are nowadays used for solving large scale linear and quadratic in many domains. In fact, the largest linear programming problem treated up to now had 1.1 billion variables and 380 million constraints, solved by Gondzio & Grothey [5] using an interior point algorithm.

The conjugate gradient algorithm (Krylov space method) has optimal performance for quadratic problems, but its extension to more general problems is not straightforward. It was superseded by quasi-Newton methods, which are more efficient for non-quadratic problems, but coincide with it in the convex quadratic case. Note that the conjugate gradient method was not motivated by the complexity study, which came later.

Accelerated gradient methods are now in the phase of development, and we are not aware of any extensive comparison with classical algorithms. Research in this field is presently very active, and it is not clear to what classes of problems this approach will be applied and which methods will be the winners in practical applications to large-scale problems.

REFERENCES

[1] AUSLENDER A & TEBOULLE M. 2006. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, **16**(3): 697–725.

[2] BECK A & TEBOULLE M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, **2**(1): 183–202, March.

[3] BIRGIN EG, MARTÍNEZ JM & RAYDAN M. 2009. Spectral Projected Gradient Methods. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, pages 3652–3659. Springer.

[4] FLETCHER R & REEVES CM. 1964. Function minimization by conjugate gradients. *Computer J.*, **7**: 149–154.

[5] GONDZIO J & GROTHEY A. 2006. Solving nonlinear financial planning problems with 10^9 decision variables on massively parallel architectures. In M. Costantino and C. A. Brebbia, editors, *Computational Finance and its Applications II, WIT Transactions on Modelling and Simulation*, 43, volume 43. WIT Press.

[6] GONZAGA CC. 1992. Path-following methods for linear programming. *SIAM Review*, **34**(2): 167–224.

[7] GONZAGA CC & KARAS EW. 2013. Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming. *Mathematical Programming*, **138**(1-2): 141–166.

[8] GONZAGA CC, KARAS EW & ROSSETTO DR. 2013. An optimal algorithm for constrained differentiable convex optimization. *SIAM Journal on Optimization*, **23**(4): 1939–1955.

[9] KARMARKAR N. 1984. A new polynomial time algorithm for linear programming. *Combinatorica*, **4**: 373–395.

[10] KHACHIYAN LG. 1979. A polynomial algorithm for linear programming. *Doklady Akad. Nauk USSR*, 244: 1093–1096. Translated in Soviet Math. Doklady 20: 191–194.

[11] LAN G, LU Z & MONTEIRO RDC. 2011. Primal-dual first-order methods with $O(1/\varepsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, **126**(1): 1–29.

[12] LUENBERGER DG & YE Y. 2008. *Linear and Nonlinear Programming*. Springer, New York, third edition.

[13] MONTEIRO RDC, ORTIZ C & SVAITER BF. 2012. An adaptive accelerated first-order method for convex optimization. Technical report, School of ISyE, Georgia Tech, July.

[14] NEMIROVSKI AS & YUDIN DB. 1983. *Problem Complexity and Method Efficiency in Optimization*. John Wiley, New York.

[15] NESTEROV Y. 2004. *Introductory Lectures on Convex Optimization. A basic course*. Kluwer Academic Publishers, Boston.

[16] NESTEROV Y. 2013. Gradient methods for minimizing composite objective function. *Mathematical Programming*, **140**(1): 125–161.

[17] NESTEROV Y & POLYAK BT. 2006. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, **108**: 177–205.

[18] NOCEDAL J & WRIGHT SJ. 2006. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, 2nd edition.

[19] POLYAK BT. 1987. *Introduction to Optimization*. Optimization Software Inc., New York.

[20] RIBEIRO AA & KARAS EW. 2013. *Otimização Contínua: aspectos teóricos e computacionais*. Cengage Learning. In Portuguese.

[21] RICHTÁRIK P. 2011. Improved algorithms for convex minimization in relative scale. *SIAM Journal on Optimization*, **21**(3): 1141–1167.

[22] ROSSETTO DR. 2012. *Tópicos em métodos ótimos para otimização convexa*. PhD thesis, Department of Applied Mathematics, University of São Paulo, Brazil. In Portuguese.

[23] SHEWCHUK JR. 1994. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, August.