

**GRADUATE SEMINAR REPORT
ON**

**INNER CONSTRUCTION
OF**

**THE APPROXIMATE SUB-DIFFERENTIAL
METHODS OF ε -Descent**

BY

BEHAILU MAMMO

Advisor

Prof. Dr. rer. nat. habil. R. Deumlich

**School of Graduate Studies
Addis Ababa University**

**June, 1998
Addis Ababa**

Sem110
S20
R5

CONTENT

ACKNOWLEDGEMENT

I owe a debt of gratitude to my instructor and advisor, Prof.Dr.rer.nat.habil.R. Deumlich for all he did in the fruition of this report.

1.2 The line-search function	8
1.3 The Ackermann algorithm	9

CHAPTER 2. Worst Implementation: Algorithms of ϵ -descent

2.1 Flattening the line-search	12
2.2 Stopping the line-search	18
2.3 The ϵ -descent algorithm and its convergence	20

REFERENCES	24
------------	----

CONTENT

	<u>Page</u>
CHAPTER 1. Identifying the Approximate Subdifferential	
1.1. The Problem and Its solution	1
1.2 The Line-Search Function	5
1.3 The Schematic Algorithm	7
CHAPTER 2. Direct Implementation: Algorithm of ϵ -Descent.	
2.1 Iterating the line-Search	12
2.2 Stopping the Line-Search	15
2.3 The ϵ -Descent Algorithm and Its convergence	20
REFERENCES	24

INNER CONSTRUCTION OF THE APPROXIMATE SUBDIFFERENTIAL: Methods of ϵ -Descent

1. Identifying the Approximate Subdifferential

1.1. The problem and its solution

In a mechanism for constructing the subdifferential of f at a given x , it has been tried to resolve the following alternatives:

- If there exists a hyperplane separating $\partial f(x)$ and $\{0\}$
i.e. a descent direction, i.e. a d with $f'(x,d) < 0$, find one.
- or, if there doesn't exist any such direction, explain why,
i.e. find a subgradient which is (approximately) 0.

This mechanism consisted in collecting information about $\partial f(x)$ in a bundle, and worked as follows:

- Given a compact convex polyhedron S with $S \subset \partial f(x)$, one computed a hyperplane separating S and $\{0\}$. This hyperplane was essentially defined by its normal vector d , interpreted as a direction, and one actually computed the best such hyperplane, namely the projection of 0 onto S . Then one made a line-search along this d , with two possible exits:
 - (a) The hyperplane actually separated $\partial f(x)$ and $\{0\}$, in which case the process was terminated. The line-search was successful and f could be improved in the direction d , to obtain the next iterate, say x_+ .
 - (b) Or the hyperplane didn't separate $\partial f(x)$ and $\{0\}$, in which case the line-search produced a new subgradient, say $s_+ \in \partial f(x)$, to improve the current S - the line-search was unsuccessful and one looped to redo it along a new direction, issuing from the same x , but obtained from the better S .

Here, two difficulties appear:

- (i) According to the whole idea of the process, the descent direction found in case (a) will be "at best" the steepest descent direction. As a result the sequence of iterates will very probably not be a minimizing sequence.

- (ii) Finding the subgradient s_+ in case (b) requires performing an endless line-search: the stepsize must tend to 0 so as to compute the directional derivative $f'(x,d)$ and to obtain s_+ . Unless f has some very specific properties (such as being piecewise affine), this can't be implemented.

In other words, the grafting will result in a non-convergent and non-implementable algorithm.

Yet the idea is good if a simple precaution is taken: rather than the purely local set $\partial f(x)$, it is $\partial_\epsilon f(x)$ that we must identify. It gathers differential information from a specific, "finitely small", neighborhood of x -called $\bar{V}_\epsilon(x)$.

Accordingly we can guess that this will fix (i) and (ii). Infact:

- (i) _{ϵ} . A direction d such that $f'_\epsilon(x,d) < 0$ is down hill not only at x but at any point in $\bar{V}_\epsilon(x)$, so a line-search along such a direction is able to derive the next iterate out of $\bar{V}_\epsilon(x)$ and the difficulty 1.1.1(i) will be solved.
- (ii) _{ϵ} . A subgradient at a point of the form $x + td$ is in $\partial_\epsilon f(x)$ for t small enough, more precisely whenever $x + td \in \bar{V}_\epsilon(x)$; so it is no longer necessary to force the stepsize in (ii) to tend to 0.

Definition 1.1.2- A nonzero $d \in \mathbb{R}^n$ is said to be a direction of ϵ -descent for f at x i.e. $f'_\epsilon(x,d) < 0$; in other words d defines a hyperplane separating $\partial_\epsilon f(x)$ and $\{0\}$.

- A point $x \in \mathbb{R}^n$ is said to be ϵ -minimal if there is no such separating d , i.e. $f'_\epsilon(x,d) \geq 0$ for all d , i.e. $0 \in \partial_\epsilon f(x)$.

Proposition 1.1.3 (i) A direction d is of ϵ -descent if and only if $f(x + td) < f(x) - \epsilon$ for some $t > 0$.

- (ii) A point x is said to be ϵ -minimal if and only if

$$f(y) \geq f(x) - \epsilon \text{ for all } y \in \mathbb{R}^n.$$

Proof: (i) Suppose d is a direction of ϵ -descent

$$\text{We recall: } f'_\epsilon(x,d) = \inf_{t>0} \frac{f(x + td) - f(x) + \epsilon}{t}$$

For given $d \neq 0$, let $\eta := f'_\epsilon(x,d)$

If $\eta < 0$ then we can find $t > 0$ such that

$$\Rightarrow f(x + td) < f(x) - \epsilon \text{ for some } t > 0.$$

Dropping (ii) Suppose x is ϵ -minimal then $0 \in \partial_{\epsilon} f(x)$

$$\Leftrightarrow f(y) \geq f(x) - \epsilon \quad \forall y \in \mathbb{R}^n \quad //$$

Now consider the following algorithmic scheme of " ϵ -descent". The descent iterations are denoted here by a superscript P .

Algorithm 1.1.4 (Conceptual Algorithm of ϵ -Descent) start from some $x \in \mathbb{R}^n$.
Choose $\epsilon > 0$ set $p = 1$

Step 1 If $0 \in \partial_{\epsilon} f(x^p)$ stop. Otherwise compute a direction of ϵ -descent, say d^p

Step 2 Make a line-search along d^p to obtain a stepsize $t^p > 0$ such that $f(x^p + t^p d^p) < f(x^p) - \epsilon$

Set $x^{p+1} = x^p + t^p d^p$. Replace p by $p + 1$ and loop to step 1

. If this scheme is implementable at all, it clearly eliminates difficulty 1.1.1(i)

Proposition 1.1.4: In algorithm 1.1.4, either $f(x^p) \rightarrow -\infty$ or the stop of step 1 occurs for some finite iteration index P^* , at which x^{P^*} is ϵ -minimal.

Proof: By construction $f(x^p) < f(x^{p-1}) - \epsilon$

$$\Rightarrow f(x^p) < f(x') - (p-1)\epsilon \quad \text{for } p = 1, 2, \dots$$

If $f(x^p) > -\infty$, we can talk about infimum:

Therefore, if f is bounded from below, say by

$\bar{f} := \inf_x f(x)$, the number P_* of iterations can't be larger than $[f(x') - \bar{f}]/\epsilon$, after which the stopping criterion of Step1 plays its role //

Hence we have eliminated the difficulty 1.1.1(i). We must now check that 1.1.1(ii) is eliminated as well.

We study one single iterations of Algorithm 1.1.4, i.e one execution of Step1, regardless of any sophisticated choice of ϵ .

In other words we are interested in the static aspect of the algorithm, in which the current iterate x^p is considered fixed.

Dropping the index P from our notation, we are given fixed $x \in \mathbb{R}^n$ and $\epsilon > 0$, and we construct compact convex polyhedra in $\partial_\epsilon f(x)$ as follows:
 . We start the process with the computation of $s(x) \in \partial f(x)$ coming from the black box (U1). We call it s , we form the polyhedron $S_1 := \{s_1\}$ and we set $k = 1$

. Recursively, suppose that $k \geq 1$ subgradients

$s_1, s_2, \dots, s_k$ with $s_j \in \partial f(x)$ for $j = 1, \dots, k$ have

been computed. They generate the compact convex polyhedron

$$S_k := \text{co} \{s_1, s_2, \dots, s_k\}$$

. If we look at Step 1 of Algorithm 1.1.4 in light of the algorithm named "Quest for a descent direction", we see that it is going to be an iterative subprocess which works as follows; we use the subscript k .

Process 1.1.5 The black box (U1) has computed a first ϵ -subgradient s_1 , and a first polyhedron S_1 is on hand:

$$s_1 := s(x) \in \partial f(x) \subset \partial_\epsilon f(x) \text{ and } S_1 := \{s_1\}.$$

- At stage k , having the current polyhedron $S_k \subset \partial_\epsilon f(x)$, compute the best hyperplane separating S_k and $\{0\}$, i.e. compute

$$-d_k = \text{Proj } 0/S_k = \text{Argmin} \{1/2 \| -d \|^2 : -d \in S_k\}$$

(recall that Proj denotes the Euclidean projection onto the closed convex hull of a set).

- Then determine whether

(a $_\epsilon$) d_k separates not only S_k but even $\partial_\epsilon f(x)$ from $\{0\}$; then the work is finished, $\partial_\epsilon f(x)$ is properly identified, d_k is an ϵ -descent direction and we can pass to the next iteration in algorithm 1.1.4

or

(b $_\epsilon$) d_k does not separate from $\{0\}$ then enlarge S_k with a new $s_{k+1} \in \partial_\epsilon f(x)$ and loop to the next k .

Ofcourse the above alternatives (a $_\epsilon$) - (b $_\epsilon$) play the same role as (a) -

(b) of 1.1.

1.2. The Line-Search Function

- Here, in order to resolve the alternatives $(a_\epsilon) - (b_\epsilon)$, and detect whether d_k is an ϵ -descent direction, we can again minimize f along d_k and see whether ϵ can thus be dropped from f .
- To this end, compute the support function $f_\epsilon(x, d_k)$ and check whether it is negative.
- This means minimizing the perturbed difference quotient defined as follows:

$$q_\epsilon(t) := \begin{cases} +\infty & \text{if } t \leq 0 \\ \frac{f(x + td) - f(x) + \epsilon}{t} & \text{if } t > 0 \end{cases} \quad (1.2.1)$$

$$\text{We are interested in } \inf_{t>0} q_\epsilon(t) \quad (1.2.2)$$

Minimizing q_ϵ is a convex problem. Via the change of variable $u = 1/t$. The function

$$u \mapsto r_\epsilon(u) := \begin{cases} q_\epsilon(1/u) = u[f(x + 1/u d) - f(x) + \epsilon] & \text{if } u > 0 \\ f'_\infty(d) = \lim_{t \rightarrow \infty} q_\epsilon(t) & \text{if } u = 0 \\ \infty & \text{if } u < 0 \end{cases} \quad (1.2.3)$$

is in $\overline{\text{Conv}} \mathbb{R}$. Its subdifferential at $u > 0$ is

$$\partial r_\epsilon(u) = \left\{ \epsilon - e(x, x + 1/u d, s) : s \in \partial f\left[x + \frac{1}{u} d\right] \right\};$$

$$\text{here } e(x, y, s) := f(x) - [f(y) + \langle s, x - y \rangle] \quad (1.2.4)$$

is the linearization error made at x when f is linearized at y with slope S

- The function r_ϵ is minimized over a non empty compact interval

$$U_\epsilon = [\underline{U}_\epsilon, \bar{U}_\epsilon], \text{ with } 0 \leq \underline{U}_\epsilon \leq U \leq \bar{U}_\epsilon < \infty$$

and its positive minima are characterized by the property

$$e(x, x + 1/u d, s) = \epsilon \text{ for some } s \in \partial f(x + 1/u d)$$

Since we live in the geometrical world of positive step sizes, we find it convenient to translate these result into the t -language.

Lemma 1.2.1. The set $T_\epsilon := \left\{ t = 1/n : n \in U_\epsilon \text{ and } n > 0 \right\}$ of minima of q_ϵ is a closed interval (possibly empty, possibly not bounded from above), which does not contain 0. Denoting by $\underline{t}_\epsilon := 1/\bar{u}_\epsilon$ and $\bar{t}_\epsilon := 1/\underline{u}_\epsilon$ the end point of T_ϵ ($1/0 = +\infty$) there holds: for any $t > 0$:

- (i) $t \in T_\epsilon \Leftrightarrow e(x, x + td, s) = \epsilon$ for some $s \in \partial f(x + td)$
- (ii) $t < \underline{t}_\epsilon \Leftrightarrow e(x, x + td, s) < \epsilon$ for all $s \in \partial f(x + td)$
- (iii) $t > \bar{t}_\epsilon \Leftrightarrow e(x, x + td, s) > \epsilon$ for all $s \in \partial f(x + td)$

. This Lemma is crucial for a one-dimensional search aimed at minimizing q_ϵ . Also, the number $\epsilon - e$, calculated at a given t , contains all the essential information of a "derivative". If it happens to be 0, t is optimal ; otherwise it indicates whether q_ϵ is increasing or decreasing near t .

Lemma 1.2.2. Suppose that d is a direction of ϵ -descent :

$$f_\epsilon'(x, d) < 0. \text{ Then } f(x + td) < f(x) - \epsilon$$

for $0 < t$ close enough to T_ϵ (i.e t large enough when $T_\epsilon = \emptyset$)

Proof: Under the stated assumption, $q_\epsilon(t)$ has the sign of its infimum, namely " $-$ ".

This fixes the (a_ϵ) -problem in process 1.1.5. As for the new s_+ , necessary in the (b_ϵ) -case, we remember that it must enjoy two properties:

- It must be an ϵ -subgradient at x . By Vertue of the transportation formula this requires precisely that q_ϵ be locally decreasing: we have

$$s \in \partial f(x + td) \text{ and } s \in \partial f(x) \text{ iff } e(x, x + td, s) \leq \epsilon \quad (1.2.5)$$

- It must separate the current S from $\{0\}$. "sufficiently well", which means it must make $\langle s_+, d \rangle$ large enough

Then the next result will be useful:

Lemma 1.2.3 Suppose that d is not a direction of ϵ -descent:
 $f'_\epsilon(x, d) \geq 0$. Then:

- (i) If $t \in T_\epsilon$, there is some $s \in \partial f(x + td)$ such that $\langle s, d \rangle \geq 0$ and $s \in \partial f(x)$

- (ii) Suppose $T_\epsilon = \emptyset$. Then

$\partial f(x + td) \subset \partial f(x)$ for all $t \geq 0$ and for all $\eta > 0$, there exists $m > 0$ such that
 $t \geq M, s \in \partial f(x + td) \Rightarrow \langle s, d \rangle \geq -\eta$

Since $f'_\epsilon(x, d) \geq 0$ then $f(x) - f(x + td) - \epsilon \leq 0$

$$\Rightarrow -t \langle s, d \rangle \leq 0 \quad \Rightarrow \langle s, d \rangle \geq 0$$

1.3 The Schematic Algorithm

We begin to see how Algorithm 1.1.4 will actually work: Step 1 will be a projection onto a convex polyhedron, followed by a minimization of the convex function r_ϵ

We restate Algorithm 1.1.4 with a some what more detailed step 1.

Also, we incorporate a criterion to stop the algorithm without waiting for the (unlikely) event " $0 \in S_k$ "

Algorithm 1.3.1 (Schematic Algorithm of ϵ -Descent)

Start from some $X' \in \mathbb{R}^n$. Choose the descent criterion $\epsilon > 0$ and the convergence parameter $\delta > 0$. Set $p = 1$

Step 1.0 (Starting to find a direction of ϵ -descent)

. Compute $s(x^p) \in \partial f(x^p)$. Set $s_1 = s(x^p)$, $k = 1$

Step 1.1 (Computing the trial direction).

. Solve the minimization problem in $\alpha \min_{\alpha \in \Delta_k} \left[1/2 \left\| \sum_{i=1}^k \alpha_i s_i \right\|^2 \right]$

Where Δ_k is the unit simplex of $\mathbb{R}^k$

Set $d_k = - \sum_{i=1}^k \alpha_i s_i$; if $\|d_k\| \leq \delta$ stop

Step 1.2 (line-search)

. Minimize γ_ϵ and conclude:

(a $_\epsilon$) either $f_\epsilon(x^p, d_k) < 0$; then go to Step 2

(b $_\epsilon$) or $f_\epsilon(x^p, d_k) \geq 0$; then obtain a suitable $s_{k+1} \in \partial_\epsilon f(x^p)$,

replace k by $k+1$ and loop to step 1.1.

Step 2 (descent)

Obtain $t > 0$ such that

$$f(x^p + td_k) < f(x^p) - \epsilon$$

Set $x^{p+1} = x^p + td_k$, replace K by $k + 1$ and loop to step 1.0

Remark - Here, at each iterate x^p , it constructs a direction d^k , along which it performs a line-search

- Two new features of this algorithm:

- First, the direction is computed in a rather sophisticated way, as the solution of an auxiliary minimization algorithm problem instead of being given explicitly in terms of the "gradient" S_k .

- The second feature brings a rather further difference that the line-search has two possible exits (a $_\epsilon$) and (b $_\epsilon$). The first case is the normal one and could be called a descent step, in which the current iterate x^p is kept where it is and the next line-search will start from the same x^p .

Now we give an idea of the convergence proof, assuming a simple situation: T_ϵ is non-empty, q_ϵ is minimized exactly at each line-search, and correspondingly, an s_{k+1} predicted by Lemma 1.2.3 is found at each null-step. Note that the last two assumptions are rather unrealistic.

We need the following fact to prove the theorem:

Fact 1. For $\epsilon \geq 0$, $\partial_\epsilon f(x)$ is a closed convex set, which is non-empty and bounded if and only if $x \in \text{int dom} f$.

Fact 2. Let $m > 0$ be fixed. Consider two sequences $\{s_k\}$ and $\{\hat{s}_k\}$, satisfying for $k = 1, 2, \dots$

$$\langle s_i, s_{k+1}, \hat{s}_k \rangle \geq m \|\hat{s}_k\|^2 \quad \text{for } i = 1, 2, \dots, k$$

If, in addition, $\{s_k\}$ is bounded, then $\hat{s}_k \rightarrow 0$ as $k \rightarrow \infty$.

Thm 1.3.2 Suppose that each execution of Step 1.2 in algorithm 1.3.1 produces an optimal $u_k > 0$ and a corresponding

$S_{k+1} \in \partial f(x^p + 1/u_k d_k)$ such that

$$0 = \epsilon - e_k \in \partial \gamma_\epsilon(U_k)$$

Where $e_k := f(x^p) - f(x^p + 1/u_k d_k) + \langle s_{k+1}, 1/u_k d_k \rangle$

Then either $f(x^p) \rightarrow \infty$, or the stop of step 1.1 occur for some finite iteration index P_x , at which there holds

$$f(X^{P^*}) \leq f(x) + \epsilon + \partial \|y - x^{P^*}\| \quad \forall y \in \mathbb{R}^n$$

Proof

Suppose $f(X^p)$ is bounded from below. As in proposition 1.1.4, Step 2 can't be executed infinitely many times

At some finite iteration index P^* , Algorithm 1.3.2 therefore loops between Step 1.1 and 1.2, i.e case (b $_\epsilon$) always occurs at this iteration. Then Lemma 1.2.3 (i) applies for each k :

First $S_{k+1} \in \partial_{\epsilon_k} f(X^{P^*}) = \partial_\epsilon f(X^{P^*})$ for all k ; we deduce in particular that the sequence

$$\{s_k\} \subset \partial_\epsilon f(X^{P^*}) \text{ is bounded} \dots \text{Fact 1}$$

Second, $\langle s_{k+1}, d_k \rangle \geq 0$ which, together with the minimality conditions for the projection $-d_k$ yields

$$\langle -d_k, s_j - s_{k+1} \rangle \geq \|d_k\|^2 \quad \text{for } j = 1, \dots, k$$

Fact 2 applies (with $\hat{s}_k = -d_k$ and $m = 1$):

$d_k \rightarrow 0$ if $k \rightarrow \infty$, so the stop must eventually occur

Finally, each $-d_k$ is a convex combination of ϵ -subgradients $s_1, \dots, s_k$ of f at X^{p*} (not that $s_1 \in \partial_\epsilon f(X^{p*})$ by construction and is therefore an ϵ -subgradient itself:

$$\begin{aligned} f(y) &\geq f(X^{p*}) - \langle d_k, y - X^{p*} \rangle - \epsilon \quad \forall y \in \mathbb{R}^n \\ \Rightarrow f(X^{p*}) &\leq f(y) + \|d_k\| \|y - X^{p*}\| + \epsilon \\ \Rightarrow f(X^{p*}) &\leq f(y) + \delta \|y - X^{p*}\| + \epsilon \quad \forall y \in \mathbb{R}^n \quad // \end{aligned}$$

2. Direct Implementation: Algorithm of ϵ -Descent

- This section contains the computational details that are necessary to implement the algorithm introduced in Chapter 1., sketched as Algorithm 1.3.1.
- These details give a good idea of the kind of questions that arise when methods for non-optimization (non-smooth optimization are implemented).
- To obtain a really implementable form of Algorithm 1.3.1, we need to specify two calculations: in Step 1.1, how the α -problem is solved, and in Step 1.2, how the line-search is performed.
- The α -problem is a classical convex quadratic minimization problem with linear constraints; as such it poses no particular difficulty.
- The line-search, on the contrary, is rather new since it consists of minimizing the non-smooth function q_ϵ or r_ϵ . It forms the subject of the present section.
- Before going to the present section, we mention line-searches.

Line Searches

- We recall a direction $d \in \mathbb{R}^n$ is said to be a direction of ϵ -descent iff $f(x + td) < f(x) - \epsilon$ for some $t > 0$
- A line-search is a sub-algorithm to find those suitable t 's such that $f(x + td) < f(x) - \epsilon$

General Structure of an algorithm

- Just as any iterative algorithm, the line-search has a rule for iteration and a stopping criterion; this means that the following two questions must be answered:
 - (i) How the sequence of trial stepsizes should be computed ?
 - (ii) When is the current trial t acceptable as the real stepsize from x_k to x_{k+1} ?
- The concept of acceptable stepsize, i.e the stopping criterion of the line-search, must be defined with great care.
- Here, it is useful to organize the stopping criterion so that it gives some indication for the next trials in case the present one is not suitable.
- The essential object characterizing a line-search is then a test which, given a stepsize t and suitable information on q_ϵ (obtained from U1), has three possibilities:
 - (O) This t is suitable and the line-search can be stopped.
 - (R) This t is not suitable and no suitable $\bar{t}$ is likely to be found on its right
 - (L) This t is not suitable and no suitable $\bar{t}$ is likely to be found on its left
- Then, any t for which (L) [resp. (R)] holds can serve as a lower [resp. upper] bound for subsequent stepsizes (trials) because this t lies to the left [resp. to the right] of the interesting area of search. As a result the following scheme suggest itself.

Algorithm A (Schematic Line-Search)

An initial trial stepsize $t > 0$ is given.

Set $t_L := 0$ and $t_R := +\infty$

- STEP 1 Test t ; in case (O) stop the line-search and pass to the next iterate x_{k+1} .
- STEP 2 In case (L) set $t_L = t$; in case (R) set $t_R = t$ Go to step 3
- STEP 3 Select a new t in (t_L, t_R) and loop to Step 1.

Remark: It is crucial that Algorithm A be finite, i.e that case (0) be reached after a finite, and if possible small, number of cycles. For this, two properties must be satisfied (taken in to consideration).

Property A_1 : (Safeguard-Reduction Property)

The update of step 3 must be such that:

- (i) infinitely many extrapolations would let $t_L \rightarrow +\infty$ and
- (ii) infinitely many interpolations would let $t_R - t_L \rightarrow 0$

Property A_2 : (Consistency Property)

The test (0), (R), (L) must be organized so that:

- (i) When t is large enough, case (L) never occurs, and
- (ii) When $t_R - t_L$ is small enough, case (0) occurs for all t in between.

Properties $A_1^{(i)}$ and $A_2^{(i)}$ taken together imply that, after a finite number of cycles, either Algorithm A stops at case (0), or it finds a $t_R < +\infty$; the number of extra polations is finite.

The (ii) combination implies likewise that the number of interpolations is finite as well. Atogether, this implies that Algorithm A must terminate.

Now to begin the main section, let us ask the following question:

Question: Given the current stepsize, assumed not suitable, how the next one can be choosen? This will be the subject of chapter 2.1.

2.1 Iterating the Line-Search

- In order to make the notation less cumbersome, when no confusion is possible, we drop the indices p and k , since they are fixed here and in the next subsection. We are therefore given $x \in \mathbb{R}^n$ and

$0 \neq d \in \mathbb{R}^n$, the starting point and the direction for the current line-search under study. We are also given $\epsilon > 0$, which is crucial in all this chapter.

- As already mentioned in Chapter 1, the aim of the line-search is to minimize the function q_ϵ , or equivalently the function r_ϵ
- We prefer to base our development on the more natural q_ϵ , despite its non convexity. Ofcourse, the reciprocal correspondence (1.2.4) must be kept in mind and used whenever necessary.

- At the point $y = x + td$, it is convenient to simplify the notation: $s(t)$ will stand for $s(x + td) \in \partial f(x + td)$ and

$$e(t) := e(x, x + td), s(t) = f(x) - f(x + td) + t \langle s(t), d \rangle \quad (2.1.1)$$

- * Recall that $\epsilon - e(t)$ is in $\partial r_\epsilon(1/t)$ (we remind the reader that the last expression means the subdifferential of the convex function $u \mapsto r_\epsilon(u)$ at the point $u = 1/t$).

- The problem to be solved in its subsection is: during the process of minimizing q_ϵ , where should we place each trial stepsize? We must decide whether a given $t > 0$ is

(O) Convenient, so the minimization of q_ϵ can be stopped.

(L) On the left of the set of convenient stepsizes

(R) On their right

- The key to designing (L) and (R) lies in the statement

$$q_\epsilon(t) := \begin{cases} +\infty & \text{if } t \leq 0 \\ \frac{f(x + td) - f(x) + \epsilon}{t} & \text{if } t > 0 \end{cases}$$

and

$$\inf q_\epsilon(t)$$

- Infact, call the black box (U1) to compute $s(t) \in \partial f(x + td)$; then compute $e(t)$ of (2.1.1) and compare it to ϵ .

- There are three possibilities:

(O) := " $\{e(t) = \epsilon\}$ " (an extraordinary event!). Then t is optimal and the line-search is finished.

(L) := " $\{e(t) < \epsilon\}$ ". This t is too small, in the sense that no optimal $\bar{t}$ can lie on it left. It can serve as a lower bound for all subsequent stepsizes, therefore set $t_L = t$ before looping to the next trial

(R) := " $\{e(t) > \epsilon\}$ ". This t is too large, so we put $t_R = t$ before looping.

Apart from the stopping criterion, we are now in a position to describe the line-search in some detail:

Algorithm 2.1.1 (Prototype Line-Search for ϵ -Descent)

STEP 0 (initialization). Set $t_L = 0$, $t_R = +\infty$. Choose an initial $t > 0$
 STEP 1 (work) Obtain $f(x + td)$ and $\delta(t) \in \partial f(x + td)$; compute $e(t)$ defined by 2.1.1
 STEP 2 (dispatching). If $e(t) < \epsilon$ set $t_L = t$; if $e(t) > \epsilon$ set $t_R = t$
 STEP 3 (stopping criterion). Apply the stopping criterion, which must include in particular step the line-search if $e(t) = \epsilon$ or if $f(x + td) < f(x) - \epsilon$
 STEP 4 (new step size). If $t_R = +\infty$ then extrapolate i.e compute a new $t > t_L$. If $t_R < +\infty$ then interpolate i.e compute a new $t \in (t_L, t_R)$

- As can be seen by a long enough contemplation of this algorithm two sequences of stepsizes $\{t_R\}$ and $\{t_L\}$ are generated. They have the properties that $\{t_L\}$ is increasing and $\{t_R\}$ is decreasing. At each cycle

$$t_L \leq \bar{t} \leq t_R$$

for any $\bar{t}$ minimizing q_ϵ .

- Once an interpolation is made i.e once some real t_R is has been found, no extrapolation is ever made again. In this case one can ascertain that q_ϵ has a minimum "at finite distance".
- The new t in step 4 must be computed so that, if infinitely many extraplations [resp. interpolations] are made, then $t_L \rightarrow +\infty$ [resp. $t_R - t_L \rightarrow 0$]. This is what was called the safeguard - reduction property A_1 .

2.2 Stopping the Line-Search

- It has been mentioned that the stopping criterion (step 3 in Algorithm 2.1.1) is by far the most important ingredient of the line-search
- Without it, no actual implementation can even be considered. It is a pivot between Algorithms 2.1.1 and 1.3.1
- As it is clear from Step 1.2 in algorithm 1.3.1, the stopping test consist in choosing three possibilites:
- either the present t is not suitable and the line-search must be continued
- or the present t is suitable because case (a_ϵ) is detected:
 $f_{\epsilon'}(x, d) < 0$; this simply amount to testing the descent enequality:

$$f(x^p + td_k) < f(x^p) - \epsilon ; \quad (2.2.1)$$
 if it is true, update x^p to $x^{p+1} := x^p + td_k$ and increase p by 1;
- or the present t is suitable because case (b_ϵ) is detected :
 $(f_{\epsilon'}(x^p, d_k) \geq 0$ and also a suitable ϵ -subgradient s_{k+1} to the current polyhedral approximation S_k of $\partial_\epsilon f(x^p)$ and increase k by 1.

Comment: Assume that (2.2.1) never happens, i.e. that $k \rightarrow \infty$ in Step 1 of Algorithm 1.3.1. Then the whole point is to realize that $0 \in \partial_{\epsilon} f(x^p)$, i.e. to have $d_k \rightarrow 0$. For this, each iteration k must produce $s_{k+1} \in \mathbb{R}^n$ satisfying two properties:

$$s_{k+1} \in \partial_{\epsilon} f(x^p) \quad (2.2.2)$$

$$\langle s_{k+1}, d_k \rangle \geq 0 \quad (2.2.3)$$

- Actually, the following specification of (2.2.3):

$$\langle s_{k+1}, d_k \rangle \geq -m' \|d_k\|^2, \quad m' \text{ fixed in } (0,1) \quad (2.2.4)$$

preserves the desired property - $d_k \rightarrow 0$.

- On the other hand, if s_{k+1} is to be the current subgradient $s(t) = s(x^p + td_k)$ ($t > 0$ being the present trial stepsize, t_L or t_R), we know from (1.2.5) that (2.2.2) is equivalent to $e(x^p, x^p + td_k, s(t)) \leq \epsilon$, i.e.

$$t \langle s(t), d_k \rangle \leq \epsilon - f(x^p + td_k) + f(x^p) \quad (2.2.5)$$

- It can be seen that the two requirements (2.2.4) and (2.2.5) are antagonistic. They may even be incompatible.

Example 2.2.1 With $n = 1$, take the objective function (2.2.6)

$$f(x) = \max \left\{ f_1(x), f_2(x) \right\} \quad \text{with} \quad \begin{cases} f_1(x) := 2 - x \\ f_2(x) := \exp x \end{cases}$$

- The solution $\bar{x}$ of $\exp x + x = 2$ is the only point where f is not differentiable. It is also the minimum of f , and $f(\bar{x}) = \exp(\bar{x})$. Note that $\bar{x} \in (0,1)$ and $f(\bar{x}) \in (1,2)$.

- The sub differential ∂f of f is

$$\partial f(x) = \begin{cases} \{-1\} & \text{if } x < \bar{x} \\ [-1, \exp \bar{x}] & \text{if } x = \bar{x} \\ \exp x & \text{if } x > \bar{x} \end{cases}$$

- Now, suppose that the ϵ -descent Algorithm 1.3.1 is initialized at $x^1 \geq 0$, where the direction of search d_1 is 1. The trace-function $t \mapsto f(0 + t.1)$ is differentiable except at a certain $\bar{t} \in [0,1)$, corresponding to $\bar{x}$.

- Let us find the linearization-error function e of (2.1.1)

$$\begin{aligned} e(t) &:= e(x, x + td, s(t)) \\ &= f(x) - f(x + td) + t \langle s(t), d \rangle \end{aligned}$$

$$\begin{aligned} \text{Now } e(t) &= f(0) - f(0 + t.1) + t \langle s(t), 1 \rangle \\ &= \max \{f_1(0), f_2(0)\} - \max \{f_1(t), f_2(t)\} + t \langle s(t), 1 \rangle \\ &= \max \{2, 1\} - \max \{f_1(t), f_2(t)\} + t \langle s(t), 1 \rangle \end{aligned}$$

If $t > \bar{t}$, then

$$\begin{aligned} e(t) &= w - \text{expt} + t \langle \text{expt}, 1 \rangle \\ &= w + (t - 1) \text{expt} \end{aligned}$$

If $0 \leq t < \bar{t}$, then

$$\begin{aligned} e(t) &= 2 - (2 - t) + t \langle -1, 1 \rangle \\ &= 0 \end{aligned}$$

Hence

$$e(t) = \begin{cases} 0 & \text{if } 0 \leq t < \bar{t} \\ 2 + (t-1) \text{expt} & \text{if } t > \bar{t} \end{cases}$$

- At $t = \bar{t}$, the number $e(t)$ is some where in $[0, e^*]$ depending on which subgradient $s(\bar{x})$ is actually computed by the black box), where

$$e^* := 3\bar{t} - \bar{t}^2 > 1$$

- Finally suppose that $\epsilon \leq e^*$, for example $\epsilon = 1$. Clearly enough,

* if $t \in [0, \bar{t})$, then $\langle s(t), d_1 \rangle = -1 = -\|d_1\|^2$ so (2.2.4) can't hold, no matter how m' is choosen.

* $t > \bar{t}$, then $e(t) > e^* \geq \epsilon$, so (2.2.5) can not hold.

- In other words, it is impossible to obtain simultaneously (2.2.4) and (2.2.5) unless the following two extraordinary events happen:

- (i) the line-search must produce the particular stepsize $t = \bar{t}$ (said otherwise, q_ϵ or r_ϵ must be exactly minimized);
 - (ii) at this $\bar{x} = 0 + \bar{t}.1$, the black box must produce a rather particular subgradient s , namely one between the two extreme points -1 and $\exp \bar{x}$, so that $\langle s, d_1 \rangle$ is large enough and (2.2.5) has a chance to hold.
- We also note that, if ϵ is large enough, namely $\epsilon \geq 2 - f(\bar{x})$, (2.2.1) can not be obtained, just by definition of $\bar{x}$.

Remark: $\bar{t}$ is given by a non-solvable equation and can not be computed exactly.

- From this example, we draw the following conclusion: it may happen that for the given f , x , d and ϵ , no line-search algorithm can produce a suitable element (new) in $\partial_\epsilon f(x)$; and this no matter how m' is chosen in $(0,1)$. When this phenomenon happens, Algorithm 2.1.1 is stuck: it loops forever between Step 1 and Step 4. Of course, relaxing condition (2.2.4), say by taking $m' \geq 1$, does not help because then, it is Algorithm 1.3.1 which may be stuck, looping within step 1: d_k will have no reason to tend to 0 and the stop of step 1.1 will never occur.
- The diagnosis is that (1.2.5), i.e the transportation formula, does not allow us to reach those ϵ -subgradient that we need. The remedy is to use a slightly more sophisticated formula, to construct ϵ -subgradient as combinations of subgradient computed at several other point:

Lemma 2.2.3 Let be given : $x \in \mathbb{R}^n$, $y_i \in \mathbb{R}^n$ and $s_i \in \partial f(y_i)$ for $j = 1, \dots, m$; set

$$e_j = e(x, y_j, s_j) = f(x) - f(y_j) - \langle s_j, x - y_j \rangle \text{ for } j = 1, \dots, m.$$

With $\alpha = (\alpha_1, \dots, \alpha_m)$ in the unit simplex Δ_m , set $e := \sum_{j=1}^m \alpha_j e_j$.

Then there holds

$$s := \sum_{j=1}^m \alpha_j s_j \in \partial_e f(x)$$

- This result can be applied to our line-search problem. Suppose that the minima of q_ϵ have been bracketed by t_L and t_R : in view of Lemma 1.2.1, we have on hand two subgradient $S_L := s(x + t_L d)$ and $s_R := s(x + t_R d)$ satisfying

$$\begin{aligned} t_L \langle s_L, d \rangle &< \epsilon - f(x) + f(x + t_L d) \\ t_R \langle s_R, d \rangle &> \epsilon - f(x) + f(x + t_R d) \end{aligned} \quad \text{and}$$

$$\text{So we have } \langle s_L, d \rangle < 0 < \langle s_R, d \rangle \quad (2.2.7)$$

Therefore, consider a convex combination

$$s^\mu := \mu s_L + (1 - \mu) s_R, \quad \mu \in [0, 1]$$

According to (2.2.7), $\langle s^\mu, d \rangle \geq -m' \|d\|^2$ if μ is small enough, namely

$\mu \leq \bar{\mu}$ with

$$\bar{\mu} := \frac{\alpha \|d\|^2 + \langle s_R, d \rangle}{\langle s_R, d \rangle - \langle s_L, d \rangle} > 0, \quad \alpha > 0 \text{ because}$$

$$\begin{aligned} \langle s^\mu, d \rangle &= \langle \mu s_L + (1-\mu)s_R, d \rangle = \mu \langle s_L, d \rangle + \langle s_R, d \rangle \\ &= \mu [\langle s_L, d \rangle - \langle s_R, d \rangle] + \langle s_R, d \rangle \\ &= -\mu [\langle s_R, d \rangle - \langle s_L, d \rangle] + \langle s_R, d \rangle \\ &\geq -\bar{\mu} [\langle s_R, d \rangle - \langle s_L, d \rangle] + \langle s_R, d \rangle \\ &= -\alpha \|d\|^2 \quad \forall \alpha > 0 \\ \Rightarrow \langle s^\mu, d \rangle &\geq -m' \|d\|^2 \end{aligned}$$

On the other hand, Lemma 2.2.3 shows that $s^\mu \in \partial_\epsilon f(x)$ if μ is large enough, namely $\mu \geq \underline{\mu}$, with

$$\underline{\mu} := \frac{e_R - \epsilon}{e_R - e_L} \in (0, 1) \quad (2.2.8)$$

It turns out (and this will follow from Lemma 2.3.2 below) that, from the property $f(x + td) \geq f(x) - \epsilon$, we have $\underline{\mu} < \bar{\mu}$ for $t_R - t_L$ small

enough. When this happens, we are done because we can find $s_{k+1} = s^\mu$ satisfying (2.2.4) and (2.2.5). Algorithm 1.3.1 can be stopped and the current approximation of $\partial_\epsilon f(x^p)$ can be suitably enlarged.

- There remains to treat the case in which no finite t_R can ever be found. Actually, this case is simpler: $t_L \rightarrow \infty$ and Lemma 1.2.3 tells us that $s(x + t_L d)$ is eventually convenient.
- A simple and compact way to carry out the calculations is to take for μ the maximal Value (2.2.8), being understood that $\mu = 1$ if $t_R = \infty$. This amounts to taking systematically $\frac{\mu}{\epsilon}$ in $\partial_\epsilon f(x)$.
- Then the stopping criterion is:
 - (a $_\epsilon$) : Stop the line-search if $f(x + td) < f(x) - \epsilon$ (and pass to the next x);
 - (b $_\epsilon$) : Stop the line-search if $\langle s^u, d \rangle \geq \alpha \|d\|^2$ (and pass to the next d).
- In all other cases, continue the line-search, i.e pass to the next trial t .
- The consistency of this stopping criterion will be confirmed in the next section.

2.3 The ϵ -Descent Algorithm and Its Convergence

We are now in a position to realize this section 2 and to write down the complete organization of the algorithm proposed. The initial iterate x^1 is on hand together with the black box (U_1) which, for each $x \in \mathbb{R}^n$, computes $f(x)$ and $s(x) \in \partial f(x)$. Further more, a number $\bar{K} \geq 2$ is given, which is the maximal number of n -vectors that can be stored, in view of the memory allocated in the computer. We choose the tolerances $\epsilon > 0, \delta > 0$ and $m' = 0.1$ is reasonable.

- The algorithm below is ofcourse a combination of Algorithms 1.3.1, 2.1.1 and of the stopping criterion introduced at the end of 2.2.2

Algorithm 2.3.1 (Algorithm of ϵ -Descent)

The initial iterate x^1 is given. Set $p = 1$.

Compute $f(x^p)$ and $s(x^p) \in \partial f(x^p)$. Set $f^0 = f(x^p)$, $s_1 = s(x^1)$

STEP 1 (Starting to find a direction for ϵ -descent). Set $k = 1$

STEP 2 (Computing the trial direction).

Set $d = \sum_{i=1}^k \alpha_i s_i$, where α solves

$$\min_{\alpha \in \Delta} \frac{1}{2} \left\| \sum_{i=1}^k \alpha_i s_i \right\|^2$$

STEP 3 (final stopping test). If $\|d\| \leq \delta$ stop.

STEP 4 (initializing the line-search). Choose an initial $t > 0$.

Set $t_L = 0$, $s_L = s_1$, $e_L = 0$; $t_R = 0$, $s_R = 0$

STEP 5 Compute $f = f(x^p + td)$ and $s = s(x^p + td) \in \partial f(x^p + td)$;

set $e = f^0 - f + t \langle s, d \rangle$

STEP 6 (dispatching). If $e < \epsilon$, set $t_L = t$, $e_L = e$, $s_L = s$

If $e \geq \epsilon$, set $t_R = t$, $e_R = e$, $s_R = s$

If $t_R = 0$, set $u = 1$, otherwise set $u = (e_R - \epsilon) / (e_R - e_L)$

STEP 7 (Stopping criterion of the line-search). If $f < f^0 - \epsilon$ go to step 11.

STEP 8 (iterating the line-search). If $t_R = 0$, extrapolate, i.e. compute a new $t > t_L$.

STEP 9 (managing the computer-memory). If $k = \bar{k}$, delete $s_1, s_2, \dots, s_k$. Insert in the new list $-d$ coming from Step 2 and let $k < \bar{k}$ be the number of elements thus obtained.

STEP 10 (iterating the direction - finding process). Set $s_{k+1} = s^\mu$. Replace k by $k + 1$ and loop to Step 2

STEP 11 (iterating the descent process). Set $x^{p+1} = x^p + td$, $s_1 = s$. Replace p by $p + 1$ and loop to Step 1.

. A loop from step 11 to step 1 represents an actual ϵ -descent with x^p moved; this was called a descent - step. A loop from step 10 to step 2 represents one iteration of the direction finding procedure, i.e. a null step one subgradient is appended to the current approximation of $\partial f(x^p)$.

- The detailed line-search is expanded between Steps 5 and 8. At each cycle, step 7 decides whether
 - to iterate the line-search, by a loop from Step 8 to 5
 - to iterate the descent process, with an actual ϵ -descent method.
 - or to iterate the descent process with an actual ϵ -descent method (obtained)

Now we have to prove convergence of this algorithm. First, we make sure that each line-search terminates.

Lemma 2.3.2 Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be convex. Suppose that the extrapolation and interpolation formulae in step 8 of Algorithm 2.3.1 satisfy the safeguard-reduction property A_1 . Then for each iteration (p, k) , the number of loops from Step 8 to Step 5 is finite.

Proof:

Suppose for some fixed iteration (p, k) , the line-search does not terminate.

Suppose first that no $t_R > 0$ is ever generated. By construction, $\mu = 1$ and $s^u = s = s_L$ for ever. One has therefore at each cycle.

$$f(x^p + t_L d) + \epsilon \geq f(x^p) \geq f(x^p + t_L d) - t_L \langle s^u, d \rangle$$

Where the first inequality holds because Step 7 never exits to step 11; the second inequality is because

$$s^u \in \partial f(x^p + t d)$$

$$\text{We deduce } t_L \langle s^u, d \rangle \geq -\epsilon \quad (2.3.2)$$

Now the assumptions on the extrapolation formulae imply that $t_L \rightarrow \infty$. In view of the test Step 7 $\rightarrow$ Step 9, (2.3.2) can not hold infinitely often. Thus some $t_R > 0$ must eventually be generated, and step 6 shows that, from then on, $\mu e_L + (1 - \mu)e_R = \epsilon$ at every subsequent cycle. This can be expressed as follows:

$$\begin{aligned} f(x^p) - \mu f(x^p + t_L d) - (1 - \mu) f(x^p + t_R d) \\ = \epsilon - t_L u \langle s_L, d \rangle - t_R (1 - u) \langle s_R, d \rangle \end{aligned}$$

Furthermore, non-exit from step 7 to step 11 implies that the left-hand side above is smaller than ϵ , so

$$\mu t_L \langle s_L, d \rangle + t_R (1-\mu) \langle s_R, d \rangle \geq 0 \quad (2.3.3)$$

By the assumption on the interpolation formulae, t_L and t_R have a common limit $\bar{t} \geq 0$, and we claim that $\bar{t} > 0$.

To show $\bar{t} > 0$, I use the following fact:

$$\left[\begin{array}{l} \text{With } f \in \text{conv} \mathbb{R}^n, \text{ let } S \text{ be a convex compact subset of } \mathbb{R}^n \text{ dom } f. \\ \text{Then there exist } L = L(S) \geq 0 \text{ such that} \\ |f(x) - f(x')| \leq L \|x - x'\| \text{ for all } x \text{ and } x' \text{ in } S \end{array} \right.$$

Suppose $\bar{t} = 0$, then taking $x = x^p$ and $x' = x^p + t_R d$, we get

$$\begin{aligned} f(x^p) - f(x^p + t_R d) &\leq L \|t_R d\| = L t_R \|d\| \\ \Rightarrow f(x^p) - f(x^p + t_R d) + t_R \langle s_R, d \rangle &\leq L t_R \|d\| + t_R \langle s_R, d \rangle \\ \Rightarrow 0 < \epsilon < e_R = f(x^p) - f(x^p + t_R d) + t_R \langle s_R, d \rangle &\leq 2L t_R \rightarrow 0 \\ \text{which is a contradiction.} \end{aligned}$$

Thus $\bar{t} > 0$ and deviding by $\bar{t} > 0$, (2.3.3) can be written as

$$\mu \langle s_L, d \rangle + (1-\mu) \langle s_R, d \rangle + \eta \geq 0 \text{ where the extra form (term)}$$

$$\eta := \frac{t_L - \bar{t}}{\bar{t}} \mu \langle s_L, d \rangle + \frac{t_R - \bar{t}}{\bar{t}} (1-\mu) \langle s_R, d \rangle$$

tends to zero when the number of cycles tends to infinity. This is impossible because of Step 7 $\rightarrow$ Step 9.

The rest is now a variant of Theorem 1.3.2

Theorem 2.3.3 The assumptions are those of Lemma 2.3.2. Then, either $f(x^p) \rightarrow -\infty$, or the stop of step 3 occurs for some finite p^* , yielding x^{p^*} satisfying the approximate minimality condition (2.3.1)

. The proof is exactly the same as Theorem 1.3.2 //

REFERENCES

- [1] Hiriart-Urvuty: Convex Analysis and Minimization
Algorithm I, Springer-Verlag, 1993
- [2] Hiriart-Urvuty: Convex Analysis and Minimization
Algorithm II, Springer-Verlag, 1993