NONSMOOTH OPTIMIZATION AND DESCENT METHODS

Claude Lemarechal*

RR-78-4
March 1978

*Institut de Recherche en Informatique et Automatique, Le Chesnay, France, and International Institute for Applied Systems Analysis.

International Institute for Applied Systems Analysis
A-2361 Laxenburg, Austria

Maria Sachs, editor
Martin Schobel, graphics

Printed by NOVOGRAPHIC
Maurer-Lange-Gasse 64
1238 Vienna

PREFACE

One of the roles of Systems and Decision Sciences at IIASA is to provide tools for studying sophisticated control systems. Accordingly, the task "Nondifferentiable Optimization" has been created to study modern methods in the field of mathematical programming, and to implement efficient minimization codes.

This paper describes the role of nondifferentiable optimization from the point of view of systems analysis, briefly describes the state of the art, and gives a new minimization method.

The author considers that this method is a first result of the Nonsmooth Optimization Workshop held at IIASA from March 28 to April 9, 1977, during which many ideas were exchanged so that the method could see the light of day.

SUMMARY

Nonsmooth optimization is a field of research actively pursued at IIASA. In this paper, we show *what* it is; a thing that cannot be guessed easily from its definition by a negative statement. Also, we show *why* it exists at IIASA, by exhibiting a large field of applications ranging from the theory of nonlinear programming to the computation of economic equilibria, including the general concept of decentralization. Finally, we show *how* it can be done, outlining the state of the art, and developing a new algorithm that realizes a synthesis between the concepts commonly used in differentiable as well as nondifferentiable optimization.

Our approach is as non-technical as possible, and we hope that a non-acquainted reader will be able to follow a non-negligible part of our development.

ABSTRACT

In Section 1, we give the basic concepts underlying nonsmooth optimization and show what it consists of. We also outline the classical methods, which have existed since 1959, aimed at optimizing nondifferentiable problems.

In Section 2, we give a list of possible applications, including acceleration of gradient type methods, general decomposition--by prices, by resources, and Benders decomposition--minimax problems, and computation of economic equilibria.

In Section 3, we give the most modern methods for nonsmooth optimization, defined around 1975, which were the first general descent methods.

In Section 4, we develop a new descent method, which is based on concepts of variable metric, cutting plan approximation and feasible directions. We study its motivation, its convergence, and its flexibility.

Nonsmooth Optimization and Descent Methods

1. BASIC CONCEPTS

1.1. The aim of "nonsmooth optimization" is to provide algorithms which minimize objective functions f whose gradient is not continuous. In such situations, the known classical methods fail to provide even a gross approximation of an optimum. This is rather obvious for the gradient type methods (steepest descent, conjugate gradient, quasi-Newton,...) but it is also true in general for methods which do not compute derivatives (Hook-Jeeves, Gauss-Seidel,...) and this fact is perhaps less well known.

1.2. Since the gradient is not continuous, there must be some points where it is not defined. However, we will suppose that f is smooth enough such that, even if the gradient does not exist at a point x, it does exist at some point $x + dx$ arbitrarily close to x. It is known that convex functions are smooth enough in this sense, and, for simplicity, we will restrict our development to the convex case.

1.3. Thus, we suppose that, given a point x, it is possible to compute the value f(x) (which is continuous) together with some vector, which we call g(x), which is either the gradient $\nabla f(x)$ if it exists, or the gradient at *some point infinitely close to* x. A simple example shows how this statement can be interpreted:

In one dimension, let f be defined by

$$f(x) = \begin{cases} 0, & \text{if } x \leq 1 \\ x^2 - 1, & \text{if } x \geq 1 \end{cases} .$$

The 1 - vector g(x) will be 0 if $x < 1$ and 2x if $x > 1$. At the point $x = 1$, there is no gradient, but we can take 0 or 2 as g(1). Thus, g(x) can, for example, be defined as

$$g(x) = \begin{cases} 0, & \text{if } x < 1 \\ 2x, & \text{if } x \geq 1 \end{cases} .$$

Note that $g(1) = 2$ is not a gradient of f anywhere, but it is the limit of $\nabla f(x)$ as $x \downarrow 1$. Of course, from the discontinuous nature of the gradient, the process for computing g(x) has to be highly unstable (small changes in x may induce large changes in g(x)). This is the reason why classical methods fail, and nonsmooth optimization is precisely aimed at eliminating this bad effect. The vector g(x), thus computed, will be called a subgradient of f at x.

To sum it up, nonsmooth optimization has nothing to do with derivative-free methods, but rather with special devices added to gradient methods for ensuring convergence.

1.4. For minimizing such a nondifferentiable function, the simplest method is the so-called "subgradient optimization", largely developed in the Soviet Union (see [15] for a review of the literature). It consists of constructing iteratively a sequence $\{x_n\}$: at each x_n, we compute $g(x_n)$ and we make a step $t_n > 0$ along the normalized direction $-g(x_n)$:

$$x_{n+1} = x_n - t_n g(x_n)/|g(x_n)| \quad .$$

Generally, the stepsize is chosen "off-line", for example:

$$t_n = \frac{1}{n} \quad \text{or} \quad t_n = t_0 \rho^n \quad ,$$

where $t_0 > 0$ and ρ is a positive number slightly smaller than 1.

Although this kind of method is quite simple (5 minutes are enough for anybody to implement it on a computer), this advantage is paid for by a serious drawback: there is no reasonable stopping criterion; one must stop the iterations when the stepsize t_n has become conveniently small, and one has no information on the optimality of x_n. Moreover, the sequence of objective values $\{f(x_n)\}$ is not monotonically decreasing. Yet, a monotone decrease of the objective would be a very nice property, which would provide at least two safeguards:

- Stability: the requirement $f(x_{n+1}) < f(x_n)$ prevents x_n to diverge or to cycle.
- Emergency stop: if the method fails for some reason (such an eventuality must unfortunately never be neglected) then one has to stop the iterations "by hand". In that case, one is at least assured of having made progress if the method is descent-wise: the last iterate is in particular better than the first one!

1.5. If it is desired to make progress at each iteration ($f(x_{n+1}) < f(x_n)$), one must spend much time computing a direction of descent because one must take into account all the possible values for $g(x_n)$. More precisely, x_n being given, construct a sequence $y_i \to x_n$ such that $\nabla f(y_i)$ has a limit (this is possible: see 1.2). Consider all such sequences $\{y_i\}$ and the set $M(x_n)$ made up of all the corresponding limits of the gradients. Note that $M(x_n)$ is just a mathematical concept, and it is generally impossible to know it explicitly.

Then it can be shown that a direction d_n issuing from x_n is a direction of descent (i.e. such that it is possible to find $t_n > 0$, such that $f(x_n + t_n d_n) < f(x_n)$) if and only if

$$(d_n, g) < 0 \quad , \text{ for all } g \in M(x_n) \quad .$$

In some special cases, studied by Demjanov [2], $M(x_n)$ is a finite set $\{g_1, \ldots, g_k\}$ that can be constructed explicitly. Then it is possible to find a descent direction, which has to satisfy a set of k inequalities. It turns out that, among all the descent directions, there is one which is particularly important: the opposite of the projection of the origin onto the convex polyhedron generated by $M(x_n)$, which plays the role of the gradient in the nondifferentiable case. This direction is therefore the solution of the quadratic program:

$$\begin{cases} \min |d|^2 \\ d = -\sum_{i=1}^{k} \lambda_i g_i \qquad g_i \in M(x_n) \\ \sum_{i=1}^{k} \lambda_i = 1 \qquad \lambda_i \geq 0 \quad . \end{cases}$$

Therefore, this program is of importance for computing a descent direction.

1.6. Another old method is the so-called cutting-plane method ([1],[7]). It is based on the following observation: from convexity, we know that

$$f(x) \geq f(y) + (g(y), x-y) \qquad \text{for all } x \text{ and } y \quad .$$

Therefore, we can write $f(x)$ under the sophisticated form

$$f(x) = \max_y \, [f(y) + (g(y), x-y)] \quad ,$$

and to minimize $f(x)$ is equivalent to minimize this max, or equivalently to solve the linear program with an infinite number of constraints (the variables are v and x):

$$\begin{cases} \min v \\ v \geq f(y) + (g(y), x-y) \ \forall y \quad . \end{cases}$$

This program cannot be solved directly, and the cutting-plane method consists in solving a sequence of linear programs with only a finite subset of constraints: when $x_1, x_2, \ldots, x_n$ have been generated together with $g(x_1), g(x_2), \ldots, g(x_n)$, one solves

$$\begin{cases} \min v \\ v \geq f(x_i) + (g(x_i), x - x_i) \quad , \quad i = 1,2,\ldots,n \quad . \end{cases}$$

One calls v_{n+1} and x_{n+1} the solution of this program. Since there are fewer constraints, it is clear that v_{n+1} is a lower bound for min $f(x)$. Then, one computes $f(x_{n+1})$ and $g(x_{n+1})$ and one solves again the linear program with $n + 1$ constraints.

Among these three methods, we consider the last two as the most interesting: 1.5. because it is the most natural extension of steepest descent, and 1.6. because it approximates convex functions by supporting hyperplanes, a technique which deserves attention. We shall use them as a basis for the methods developed in Sections 3 and 4.

2. FIELD OF APPLICATION FOR NONDIFFERENTIABLE OPTIMIZATION

The first question we must answer is: is it really so important to study algorithms for nondifferentiable optimization, or is it only a mathematical sport? Actually, such algorithms have a rather large field of application.

2.1. The class of objectives we are interested in can be approximated by smooth functions (for example polynomials) which can be minimized by classical methods. However, when the approximation becomes tighter, the minimization becomes slower, and one might try to accelerate it.

Saying it another way, nondifferentiable optimization can be viewed as a study of accelerating classical methods for stiff problems. This is a very important application.

2.2. It seems that the main field of application should be the general decomposition problem. Suppose we have to solve a large-scale decomposable problem

$$\begin{cases} \min \sum_{i=1}^{m} f_i(x_i) \\ \sum_{i=1}^{m} g_i(x_i) \leq b \end{cases} \tag{1}$$

in which the vector b can be considered as resources to be shared between the local units indexed by i. On way of decomposing this problem is to attribute a price of consumption λ to the resource: a local unit consuming $g_i(x_i)$ has to pay $(\lambda, g_i(x_i))$ so that its own objective becomes

$$\min f_i(x_i) + (\lambda, g_i(x_i)) \quad . \tag{2}$$

Call $h_i(\lambda)$ the optimum value of this program. The coordination problem is then to find the optimal λ, i.e. such that the solutions of (2) make up a solution of (1).

Duality theory says that such prices exist if (1) is convex and they solve the nonsmooth optimization problem:

$$\max \sum h_i(\lambda) - (\lambda, b) = h(\lambda) \quad .$$

Decomposition theory is extensively studied in the literature. See for example [9] for a comprehensive exposition.

Note that the index i might be the time, i.e. (1) might be a dynamic problem, in which x_i is the decision vector to be made at time i and b is the total resource available over the planning horizon $\{1,2,\ldots,m\}$ (cf. [17]).

2.3. Another method for solving (1) is the so-called "right-hand side" decomposition ([4],[6],[20]). The resource b can be shared into "quotas" $y_1,\ldots,y_m$, such that $\Sigma\, y_i = b$, which are attributed to the local units. Each local unit i is then not allowed to consume more than y_i, and its own optimization problem becomes

$$\begin{cases} \min f_i(x_i) \\ g_i(x_i) \le y_i \quad . \end{cases} \tag{3}$$

Call $v_i(y_i)$ the optimum value in (3). As in 2.2, the coordination problem is to find the optimal quotas, such that the solutions of (3) make up a solution of (1). It can be shown rather easily that this consists of solving the nonsmooth problem

$$\begin{cases} \min \sum_{i=1}^{m} v_i(y_i) = v(y) \\ \sum y_i = b \quad . \end{cases}$$

In these two examples 2.2 and 2.3, computing the value and the subgradient of the objective function ($h(\lambda)$ in 2.2, $v(y)$ in 2.3) amounts to solving m local problems, which might be a rather long process. This justifies seeking a sophisticated method, which carefully uses the information given by this process.

2.4. More generally, one may have to solve an ordinary optimization problem in which a natural grouping of the variables appears. Consider for example the problem

$$\min_{x,y} c(x,y)$$

and suppose that, fixing x, the minimization with respect to y alone is very simple (for example, if c is linear in y).

It is then desirable to strive to solve min f(x), where f(x) is the function

$$f(x) = \min_{y} c(x,y) \quad .$$

Here again, f(x) is in general not differentiable, and we can justify this statement intuitively. If the minimizer y(x) is unique, one has f(x) = c(x,y(x)). Then we can write formally

$$\frac{df(x)}{dx} = \frac{\partial}{\partial x} c(x,y(x)) + \frac{\partial}{\partial y} c(x,y(x)) \frac{d}{dx} y(x) \quad .$$

Now, since y(x) is optimal one has $\frac{\partial c}{\partial y} = 0$. Therefore, when the minimizer y(x) is unique, one has in general $g(x) = \frac{df}{dx} = \frac{\partial}{\partial x} c(x,y(x))$.

When it is not unique, there are several "gradients", i.e. no gradient at all. Computing M(x) (defined in 1.5.) consists in that case in finding all the solutions of $\min_y c(x,y)$.

Such a technique is known as the Benders decomposition. It has been applied in [5] for mixed integer programming problems, using algorithms of the type 1.6.

2.5. Some problems can be encountered in which the objective function has the so-called minimax form:

$$f(x) = \max_{i=1}^{m} f_i(x) \quad .$$

Again, there is no gradient at points x such that the max is obtained for several values of i. As a good example, we can mention the problem of finding an economic equilibrium [8].

For i = 1,2,...,n let $Z_i(x)$ be n functions, called the excess demands, depending on the price x. Generally, these functions have the property that, for each x, there is at least one $Z_i(x)$ which is nonnegative, and there exists an $\bar{x}$ such that the $Z_i(\bar{x})$ are all zero. Such an $\bar{x}$, called an economic equilibrium, must be found.

In some examples, $Z_i(x)$ are multivalued functions, and in that case, there exists only one algorithm [19] for finding an equilibrium. It is combinatorial in nature, and its computational efficiency is much debated.

On the other hand, when $Z_i(x)$ are well-defined continuous functions, it might be more interesting to have "descent" methods in which the excess demand is reduced at each iteration.

This can be done by defining the function

$$f(x) = \max_{i=1}^{n} \alpha_i Z_i(x)$$

where the positive coefficients α_i are suitably chosen. This function is always positive; its minimum is zero, obtained at an equilibrium $\bar{x}$. Minimizing $f(x)$ is a nonsmooth optimization problem (note that we do not suppose that $Z_i(x)$ is continuously differentiable).

3. METHODS OF DESCENT

3.1. Most classical algorithms of minimization determine the new iterate x_{n+1} by computing first a direction d_n issued from x_n, and then a positive stepsize t_n.

For computing the stepsize, a technique has been recently developed in [21], which seems quite satisfactory; we describe it now.

In addition to x_n and d_n, one has on hand a negative number q. Considering the univariate function $h(t) = f(x_n + td_n)$, defined for $t \geq 0$, q is generally an estimate of $h'(0)$: $q = (d_n, g_n)$.

Then two numbers m_1 and m_2 such that $0 < m_2 < m_1 < 1$ are chosen. They are generally fixed throughout the algorithm; $m_1 = 0.2$, $m_2 = 0.1$ is a reasonable choice. The stepsize t_n is sought, with $x_{n+1} = x_n + t_n d_n$, and $g_{n+1} = g(x_{n+1})$ satisfying two requirements:

(a) $(d_n, g_{n+1}) \geq m_1 q$;

(b) $f(x_{n+1}) \leq f(x_n) + m_2 t_n q$.

Recalling the interpretation of q, (a) means that the new gradient is sufficiently different from the old one, and (b) means that the objective has sufficiently decreased.

It can be shown that these two requirements are consistent, provided $q \geq h'(0)$. If this does not hold (which might be the case when x_n is a point of nondifferentiability) then (b) might be impossible to obtain with $t_n > 0$.

In order to prevent this case, one must choose an additional tolerance $\varepsilon > 0$ and look for t_n such that (a) is satisfied together with

$$(b') \quad f(x_{n+1}) - t_n(g_{n+1},d_n) \geq f(x_n) - \varepsilon \quad .$$

It can be shown also that, when (b) is impossible, then (a) and (b') are satisfied by any t_n small enough. The interpretation of ε is given by observing that the term $f(x_{n+1}) - t_n(g_{n+1},d_n)$ is the value at $y = x_n$ of the linear function $f(x_{n+1}) + (g_{n+1},y - x_{n+1})$. This is the approximation of the convex function f, linearized at x_{n+1}. Thus, (b') holds when the approximation agrees with the actual $f(x_n)$ to a precision of at least ε.

When (b') holds with a small ε, this means that g_{n+1} is almost in $M(x_n)$. Therefore, this g_{n+1} should be taken into account when defining a descent direction (which should satisfy $(g_{n+1},d) < 0$) (see 1.5). As a consequence, when the line search fails to meet (b), and gives (a)-(b'), a new direction is computed from x_n, taking into account the new information g_{n+1}.

3.2. The problem of computing the direction properly is not so easy, and we shall investigate it now. It is commonly admitted that this computation should make use of the information $x_1,x_2,\ldots,x_n$ and $g_1,g_2,\ldots,g_n$ accumulated during the previous iterations, which must be memorized in one way or another. All classical accelerating devices do that.

A method which has proved rather efficient, called the "conjugate subgradient method", proceeds as follows: according to some selection rule, extract a subset $I_n \subset \{1,2,\ldots,n\}$. Then define the finite set $G_n = \{g_i \mid i \in I_n\}$.

The method computes the direction d_n by finding the projection of the origin onto G_n. In other words, one solves the quadratic program

$$\left\{\begin{array}{l} \min |d|^2 \\ d = - \sum\limits_{i \in I_n} \lambda_i g_i \\ \lambda_i \geq 0 \quad \sum \lambda_i = 1 \quad . \end{array}\right.$$

Then, the stepsize is computed by the line search of Section 3.1, where $q = -|d_n|^2$; the tolerance ε is a convergence parameter, and when d_n is zero, then convergence is obtained within $\varepsilon : f(x_n) \leq \min f + \varepsilon$ (provided the selection rule is also based on the use of ε).

To interpret and define the selection rule, we can compare with the algorithm of Section 1.5. We see that G_n is supposed to approximate $M(x_n)$. Therefore, one should select those g_i such that x_i is close to x_n, to a degree related to ε. Convergence has been proved for various selection rules ([11],[14],[21]). The algorithm has been encoded as a FORTRAN program, implemented in particular at IIASA on the PDP 11 computer. It is currently used for example to compute economic equilibria in the international models for food and agriculture.

This program is rather easy to use, with respect to its degree of generality. The user has to write a subprogram which computes f(x) and g(x), and to define some tolerances. The algorithm is quite fail-safe, and some safeguards have been incorporated to take care of nonconvexity. However, we must say that it is rather slow, in particular if some of the tolerances are not carefully chosen.

3.3. The conjugate gradient method has a certain lack of flexibility, which partly explains its modest performances: for each g_i, the only choice is to discard it or to incorporate it into G_n. No possibility is allowed to weight it. Yet, in view of Section 1.5., the projection of the origin onto G_n has a meaning only if $G_n = M(x_n)$.

Therefore, it should be better to use some approximation of $M(x_n)$ which is not simply a subset of $\{g_1, g_2, \dots, g_n\}$. It appears that one can define the n positive numbers

$$\alpha_i = f(x_n) - [f(x_i) + (g_i, x_n - x_i)] \tag{4}$$

and consider the convex polyhedron

$$G_n(\varepsilon) = \{g = \sum_{i=1}^{n} \lambda_i g_i \mid \lambda_i \geq 0, \sum \lambda_i = 1, \sum \lambda_i \alpha_i \leq \varepsilon\}$$

where ε is some positive number. By suitably adjusting ε, one can make $G_n(\varepsilon)$ approximate $M(x_n)$. One is then led to the "bundle method" defined in [12]: the direction d_n is computed as the solution of the quadratic program

$$\begin{cases} \min |d|^2 \\ d = - \sum \lambda_i g_i \\ \sum \lambda_i = 1, \ \lambda_i \geq 0, \sum \lambda_i \alpha_i \leq \varepsilon \end{cases} .$$

The stepsize is then computed as in 3.1 and 3.2.

It can be shown that this method is closely related to the method of Section 1.6. Thus, it realizes a synthesis between the descent methods of the type 1.5, and cutting-plane methods (which are not descent). It has been imolemented as an experimental FORTRAN orogram. Its performances aopear to be very sensitive to the choice of ε, which unfortunately is difficult to choose. In fact, one needs more information such as curvature to guess its prooer value.

4. A NEW METHOD

The design of a method makes use of a "model" of the objective function. For example, a conjugate gradient or quasi-Newton method uses a quadratic model, i.e. it supposes that the objective looks like a quadratic. On the contrary, a cutting-plane method supposes that it looks like a piecewise linear function. Of course, the performance of a given method is likely to depend on how the actual objective fits into the chosen model. Curiously enough, it has been observed experimentally that a quadratic model is generally a rather good representation, even of piecewise linear functions. This justifies our next development, in which we adopt the strategy: try to use a quadratic model as long as it does not deviate too much from the actual objective.

4.1. Motivation

Let us denote d the movement from x_n : $x_{n+1} = x_n + d$. Suppose that a symmetric positive definite matrix A_n is given, so that the value of the objective $f(x_n + d)$ can be approximated by the quadratic function

$$f(x_n) + (g_n,d) + \frac{1}{2}(d,A_n d) \quad . \tag{5}$$

(If the objective were a quadratic, A_n should be its hessian).

On the other hand, the cutting-plane relations give n lower bounds on the predictable value $f(x_{n+1})$:

$$f(x_{n+1}) = f(x_n + d) \geq f(x_i) + (g_i, x_n + d - x_i) \quad ,$$

$$i = 1,\dots,n \quad .$$

This can be arranged as

$$f(x_n + d) \geq f(x_n) - [f(x_n) - f(x_i) - (g_i, x_n - x_i)] + (g_i, d)$$

or, using the definition (4):

$$f(x_n + d) \geq f(x_n) - \alpha_i + (g_i, d) \quad .$$

Therefore, if we want $f(x_n + d)$ to be strictly lower than $f(x_n)$, it is absolutely necessary that d satisfy

$$-\alpha_i + (g_i, d) < 0 \qquad i = 1, \dots, n \quad . \tag{6}$$

In (6), each term $-\alpha_i + (g_i, d)$ represents the best possible decrease from $f(x_n)$ to $f(x_n + d)$; they must be strictly negative. It is therefore convenient to look for a d which makes a balance between diminishing the approximation (5) as much as possible, while keeping all the lower bounds (6) as small as possible.

We think it reasonable to take d which solves the following program in d and v:

$$\left\{\begin{array}{l} \min\ v + \frac{1}{2}\,(d, A_n d) \\ -\alpha_i + (d, g_i) \leq v \quad , \qquad i = 1, \dots, n \quad . \end{array}\right. \tag{7}$$

4.2. Justification

This program is closely related to the direction-finding problem of Pshenichnyi [18], and to the boxstep method [13] (in which one would take a "box" of the form $(d, A_n d) \leq t$). It can be partly justified by some heuristic considerations.

Since $\alpha_i \geq 0$, the point $d = 0$, $v = 0$ is feasible in (7) and the optimum d_n, v_n must satisfy $v_n + \frac{1}{2}\,(d_n, A_n d_n) < 0$. (If it were 0, x_n would be optimal.) Since A_n is positive definite, this implies $v_n < 0$ and

$$-\alpha_i + (g_i, d_n) \leq v_n < 0 \quad .$$

Also, since $\alpha_n = 0$, we have

$$(g_n,d_n) + \frac{1}{2}(d_n,A_nd_n) \leq v_n + \frac{1}{2}(d_n,A_nd_n) < 0 \quad .$$

Thus, d_n at least makes the balance mentioned above, in some sense. Also, it seems reasonable to suppose that the last constraint $(d,g_n) \leq v$ is active in the optimum (otherwise d_n would not depend on g_n), so that (7) is just a disguised way to write

$$\begin{cases} \min\ (g_n,d) + \frac{1}{2}(d,A_nd) \\ -\alpha_i + (g_i,d) \leq (g_n,d) \quad , \quad i = 1,\dots,n-1 \quad , \end{cases}$$

which we do not like because, in nonsmooth optimization, there is no reason to particularize g_n (for example, it has no meaning if f is a piecewise linear function, whose x_n is a vertex). However, its interpretation is clear: we require that the lower bounds (6) be not tighter than the classical first-order approximation, and the freedom left for d is used to minimize the quadratic approximation (5).

4.3. The Algorithm

Once d_n is computed by (7), one should make a line search producing either (a) and (b), or (a) and (b'). From the discussion in Section 4.2 ($h'(0)$ is (g_n,d_n), which is v_n), it is convenient to take $q = v_n$ (the notations are those of Section 3.1). However, we do not see exactly how to choose ε. Moreover, convergence is helped if t_n is bounded from below. Therefore, we will make the line searches as follows.

Try first $t = 1$. If $f(x_n + d_n) \leq f(x_n) + m_2v_n$, then (b) is satisfied, and we *extrapolate* to find $t_n \geq 1$ satisfying (a),(b). This will be called a *serious step;* x_n,A_n and the α_i will be updated. For updating A, we choose the fashionable BFGS formula since (a) preserves positive definiteness [3].

If $f(x_n + d_n) > f(x_n) + m_2 v_n$, then we will recompute a new direction issued from the same x_n by adding in (7) a new constraint

$$-\alpha^+ + (g^+, d) \leq v$$

where $g^+ = g(x_n + d_n)$ and α^+ is computed as in (4), namely

$$\alpha^+ = f(x_n) - [f(x_n + d_n) + (g^+, x_n - x_n - d_n)] \quad ,$$

i.e.

$$\alpha^+ = f(x_n) - f(x_n + d_n) + (g^+, d_n) \quad . \tag{8}$$

This will be called a *null-step*.

For solving (7), it is convenient to consider its dual, which gives useful information. The Lagrange function is $L(v,d,\lambda) = \frac{1}{2}(d, A_n d) + (d, \Sigma\lambda_i g_i) - \Sigma\lambda_i\alpha_i + (1 - \Sigma\lambda_i)v$. It is defined for $\lambda \geq 0$. The dual function, $h(\lambda) = \min_{v,d} L(v,d,\lambda)$, is defined for $\sum \lambda_i = 1$ (otherwise it is $-\infty$) and is then obtained for

$$d(\lambda) = -A_n^{-1} \sum \lambda_i g_i \quad . \tag{9}$$

Thus, (7) is solved by this $d(\lambda)$, where λ maximizes the dual function, namely:

$$\left\{ \begin{array}{l} \min \frac{1}{2} (\sum \lambda_i g_i, \sum \lambda_i A_n^{-1} g_i) + \sum \lambda_i \alpha_i \\ \sum \lambda_i = 1 \quad , \qquad \lambda_i \geq 0 \quad . \end{array} \right. \tag{10}$$

To recover v_n, we can write that the dual and primal values are equal, i.e.

$$\frac{1}{2}(d_n, A_n d_n) + v_n + \frac{1}{2}(\sum \lambda_i g_i, \sum \lambda_i A_n^{-1} g_i) + \sum \lambda_i \alpha_i = 0 \quad ,$$

which because of (9) can be written as

$$v_n = -(d_n, A_n d_n) - \sum \lambda_i \alpha_i \quad . \tag{11}$$

Thus, v_n is a convergence parameter, supposed to converge to zero, which, when it is small, provides an approximate optimality condition given by the following result:

Theorem 1: If $v_n \geq -\varepsilon$, *and if* $(A_n^{-1} Z, Z) \geq c|Z|^2, \forall Z$, *then one has*

$$\forall y: \quad f(y) \geq f(x_n) - \sqrt{\varepsilon/c}\ |y - x_n| - \varepsilon \quad .$$

Proof: From convexity and (4)

$$\forall y: \quad f(y) \geq f(x_n) + (g_i, y - x_n) - \alpha_i$$

which, by convex combination, gives

$$f(y) \geq f(x_n) + (\sum \lambda_i g_i, y - x_n) - \sum \lambda_i \alpha_i \quad ,$$

where λ solves (10).

Now v_n is composed of two negative terms so that $v_n \geq -\varepsilon$ implies $(d_n, A_n d_n) \leq \varepsilon$ and $\sum \lambda_i \alpha_i \leq \varepsilon$.

Now set $s = A_n d_n = -\sum \lambda_i g_i$.

The positive definiteness of A_n^{-1} can be written:

$$(d_n, A_n d_n) = (s, A_n^{-1} s) \geq c|s|^2 = c|\sum \lambda_i g_i|^2$$

so that $|\Sigma \lambda_i g_i| \leq \sqrt{\varepsilon/c}$.

Finally, by applying the Cauchy-Schwarz inequality to $(\Sigma \lambda_i g_i, y - x_n)$, we obtain:

$$\forall y: \quad f(y) \geq f(x_n) - \sqrt{\varepsilon/c}\ |y - x_n| - \varepsilon \quad . \quad \text{Q.E.D.}$$

We can now state an algorithm extending quasi-Newton methods to nondifferentiable objective functions:

```
Step 1

    $x_n$; $g_1,\dots,g_n$; $\alpha_1,\dots,\alpha_n$ are given; $\varepsilon > 0$ is a tolerance.
    $H_n = A_n^{-1}$ is a quasi-Newton matrix ($H_1$ = the identity matrix).
    Solve (10) for $\lambda$ and obtain $d_n$ and $v_n$ by (9) and (11).
    If $v_n \geq -\varepsilon$ STOP.

Step 2

    Compute $f(x_n + d_n)$ and $g_{n+1} = g(x_n + d_n)$.
    If $f(x_n + d_n) > f(x_n) + m_2 v_n$, then:
    Set $x_{n+1} = x_n$.
    Compute $\alpha_{n+1} = f(x_n) - f(x_n + d_n) + (g_{n+1}, d_n)$.
    Increase n by 1 and go to 1.

Step 3

    Otherwise, extrapolate to find $t_n \geq 1$ and $g_{n+1} = g(x_n + t_n d_n)$
    such that
        $\begin{cases} f(x_n + t_n d_n) \leq f(x_n) + m_2 t_n v_n \\ (g_{n+1}, d_n) \geq m_1 v_n \end{cases}$ .
    Set $x_{n+1} = x_n + t_n d_n$.
    Change $\alpha_i$ to $\alpha_i + f(x_{n+1}) - f(x_n) - (g_i, x_{n+1} - x_n)$   $i = 1,\dots,n$
      and set $\alpha_{n+1} = 0$.
    Update $H_n$, for example by the well-known Broyden-Fletcher-
    Goldfarb-Shanno formula [3]. Save $g_{n+1}$ for possible subse-
       quent update of H.
    Increase n by 1 and go to 1.
```

4.4. Convergence

According to Theorem 1, there are two independent properties which ensure $\{x_n\}$ to be a minimizing sequence for f. One is that a subsequence of v_n tends to zero, in order that the STOP in Step 1 eventually occurs. The second is that H_n remains uniformly positive definite, so that the optimality condition holds.

This second property depends just on studies in quasi-Newton methods. Thanks to the m_1 requirement and the fact that $v_n < -\varepsilon$ and $t_n \geq 1$, it seems that it should hold (cf. [16]). Therefore we will study here the first property only.

Theorem 2: There cannot be an infinite number of serious steps unless $f(x_n') \to -\infty$.

Proof (straightforward): Between two consecutive serious steps, say n and p, we have

$$f(x_p) \leq f(x_n) + m_1 v_{p-1} \quad .$$

If $v_{p-1} < -\varepsilon$ for all serious steps, $f(x_p)$ goes to $-\infty$. Q.E.D.

Theorem 3: There cannot be an infinite number of null-steps, unless g_n *is unbounded.*

Proof: We follow [10, Theorem 2.3], by proving first that d_n and v_n are bounded, and then that there is a subsequence of v_n which tends to zero.

It is very important to note that, when a series of null-steps is built up, H_n and every α_i are *fixed.*

It is clear that, if $|d_n| \to +\infty$, then $\frac{1}{2}(d_n, A_n d_n) \to +\infty$ like $|d_n|^2$, whereas v_n cannot go to $-\infty$ faster than $|d_n|$. Therefore the optimal value in (7) cannot remain negative.

Hence, d_n is bounded, and v_n is bounded from below.

Now take a subsequence such that $d_n \to \bar{d}$ and $v_n \to \bar{v}$. It is clear that $\bar{v} \leq -\varepsilon$. Let n and p, $p > n$ be two consecutive indices of this subsequence. From the feasibility of d_p and v_p, we have that

$$-\alpha_{n+1} + (g_{n+1}, d_p) \leq v_p \quad . \tag{12}$$

Now when executing Step 2, we have

$$f(x_n) - f(x_n + d_n) < -m_2 v_n$$

which, together with the definition of $\alpha_{n+1} = f(x_n) - f(x_n + d_n) + (g_{n+1}, d_n)$, implies

$$\alpha_{n+1} - (g_{n+1}, d_n) < -m_2 v_n \quad . \tag{13}$$

Adding (12) and (13) yields

$$(g_{n+1}, d_p - d_n) < v_p - m_2 v_n \quad .$$

Pass to the limit: if g_{n+1} is bounded, the left-hand side tends to zero, and

$$0 \leq (1 - m_2)\bar{v} \quad .$$

Since $m_2 < 1$, $\bar{v} \geq 0$ which contradicts $\bar{v} \leq -\varepsilon$. Q.E.D.

It is worth mentioning that these two proofs are independent of the chosen formula for updating H_n. In other words, the algorithms terminate at some point provided that each H_n is positive definite. It is only for getting optimality condition at this point that the uniform positive definiteness of H_n is required.

4.5. Variants

In the dual form (10) of the direction finding problem, the linear term $\Sigma\ \lambda_i \alpha_i$ can be considered as the dualization of a constraint of the form $\Sigma\ \lambda_i \alpha_i \leq \varepsilon$, for some ε. This means that there exists $\varepsilon \geq 0$ such that (10) is equivalent to

$$\min \frac{1}{2} \left(\sum \lambda_i g_i \quad , \quad \sum \lambda_i H_n g_i\right)$$
$$\sum \lambda_i = 1 \quad , \quad \lambda_i \geq 0 \quad , \quad \sum \lambda_i \alpha_i \leq \varepsilon \quad ,$$

which appears to be strongly related to the bundle methods of Section 3.3. We can actually show that our present method is a form of boxstep method, in which the box is chosen according to the norm induced by A_n (instead of the Euclidean norm, as in 3.3, or a linear norm as in [13]). In such a method, the direction d is the solution of

$$\begin{cases} \min v \\ v \geq f(x_n) - \alpha_i + (g_i,d) \\ (d,A_n d) \leq t \end{cases}$$

and proceeding as in [12], we can show that this gives the solution of (7), provided that t is suitably chosen. This observation suggests that the role of A_n is relatively minor and it might suffice to consider for example a diagonal matrix. It would be very worthwhile since the present form with a full matrix requires a large amount of data.

Another modification concerns the line search: the reason why we have given up a complete line search is that, in case of a serious step, t_n must be bounded from below (cf. the proof of Theorem 2) and, in case of a null-step, $t_n = 1$ suffices to provide

$$-\alpha_{n+1} + (g_{n+1},d_n) \geq m_2 v_n$$

which is the key argument for proving Theorem 3.

However, requiring $t_n \geq 1$ for a serious step might be too severe. Particularly at the beginning of the algorithm, when H_n is not yet properly updated, one might have to make many null-steps, which do not diminish the objective, and do not update H_n. Therefore, it might be wise to allow for smaller values for t_n (for example $t_n \geq 0.1$) by modifying Step 2 of the algorithm: when $f(x_n + d_n) > f(x_n) + m_2 v_n$, we test $f(x_n + 0.1\ d_n) > f(x_n) + 0.1\ m_2 v_n$. If it is true, we compute $\alpha_{n+1} = f(x_n) - f(x_n + 0.1\ d_n) + 0.1\ (g(x_n + 0.1\ d_n),d_n)$ and go to Step 1. If it is false, we determine a serious step $t_n \in [0.1,1]$.

The proof of Theorem 2 still holds, and we now show that the proof of Theorem 3 also holds.

Theorem 4: Let $t \in]0,1]$. *Suppose* $f(x_n + td_n) > f(x_n) + tm_2 v_n$, *and denote* $g = g(x_n + td_n)$, $f = f(x_n + td_n)$. *Set* $\alpha = f(x_n) - f + t(g,d_n)$. *Then*

$$-\alpha + (g,d_n) > m_2 v_n \quad .$$

Proof: By hypothesis, we have

$$-\alpha + (g,d_n) = f - f(x_n) - t(g,d_n) + (g,d_n)$$

$$> tm_2v_n + (1-t)(g,d_n) \quad .$$

Now by convexity, we know that $f(x_n) \geq f - t(g,d_n)$; therefore

$$(g,d_n) \geq \frac{f-f(x_n)}{t} > m_2v_n \quad .$$

We conclude that

$$-\alpha + (g,d_n) > tm_2v_n + (1-t)m_2v_n = m_2v_n \quad . \qquad \text{Q.E.D.}$$

REFERENCES

[1] Cheney, E.W., and A.A. Goldstein, Newton's Method for Convex Programming and Chebyshev Approximation, *Numerische Mathematik*, 1, 1 (1959), 253-268.

[2] Demjanov, V.F., Algorithms for Some Minimax Problems, *Journal of Computer and Systems Sciences*, 2 (1968), 342-380.

[3] Fletcher, R., A New Approach to Variable Metric Algorithms, *The Computer Journal*, 13, 3 (1970), 317-322.

[4] Geoffrion, A.M., Primal Resource-Directive Approaches for Optimizing Nonlinear Decomposable Systems, *Operations Research*, 18, 3 (1970), 375-403.

[5] Geoffrion, A.M., Generalized Benders Decomposition, *Journal of Optimization Theory and Application*, 10, 4 (1972), 237-260.

[6] Hogan, W.W., Directional Derivatives for Extremal-Valued Functions with Applications to the Completely Convex Case, *Operating Research*, 21, 1 (1973), 188-209.

[7] Kelly, J.E., The Cutting Plane Method for Solving Convex Programs, *Journal of the Society for Industrial and Applied Mathematics*, 8 (1960), 703-712.

[8] Keyzer, M., *Linking National Models of Food and Agriculture: An Introduction*, RM-77-2, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1977.

[9] Lasdon, L.S., *Optimization Methods for Large Scale Problems*, Macmillan, London, 1970.

[10] Lemarechal, C., An Algorithm for Minimizing Convex Functions, in: J.L. Rosenfeld, ed., *Information Processing '74*, North-Holland, Amsterdam, pp. 552-556.

[11] Lemarechal, C., An Extension of Davidon Method to Nondifferentiable Problems, in M.L. Balinski and P. Wolfe, eds., *Nondifferentiable Optimization*, Mathematical Programming Study 3, North-Holland, Amsterdam, 1975, pp. 95-109.

[12] Lemarechal, C., Combining Kelley's and Conjugate Gradient Methods, in *Abstracts, IX International Symposium on Mathematical Programming*, Budapest, 1976, pp. 158-159.

[13] Marsten, R.E., et al., The Boxstep Method for Large-Scale Optimization, *Operations Research*, 23, 3 (1975), 398-405.

[14] Mifflin, R., An Algorithm for Constrained Optimization with Semismooth Functions, *Mathematics of Operations Research*, 2, 2 (1977), 191-207.

[15] Poljak, B.T., Subgradient Methods (A Survey of Soviet Research), in C. Lemarechal and R. Mifflin, eds., *Nonsmooth Optimization*, Conference Proceedings, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1978 (in press).

[16] Powell, M.J.D., *Some Global Convergence Properties of a Variable Metric Algorithm for Minimization without Exact Line Searches*, Working Paper, CSS-15, Atomic Energy Research Establishment, Harwell, Didcot, Berks., 1975.

[17] Propoi, A.I., *Dual Systems of Dynamic Linear Programming*, RR-77-9, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1977.

[18] Pshenichnyi, B.N., and Yu.M. Danilin, *Numerical Methods for Extremum Problems*, Nauka, Moscow, 1975 (in Russian).

[19] Scarf, H., The Approximation of Fixed Points of a Continuous Mapping, *SIAM Journal of Applied Mathematics*, 15, 5 (1967), 1238-1342.

[20] Silverman, G.J., Primal Decomposition of Mathematical Programs by Resource Allocation, *Operations Research*, 20, 1 (1972), 28-33.

[21] Wolfe, P., A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions, in M.L. Balinski and P. Wolfe, eds., *Nondifferentiable Optimization*, Mathematical Programming Study 3, North-Holland, Amsterdam, 1975, pp. 145-173.

RELATED IIASA PUBLICATIONS

Analytical Studies of the Hurst Effect: A Survey of the Present Position. A.A. Anis, E.H. Lloyd. (RR-75-029) $1.50 AS30.

Generalized X-Y Functions, the Linear Matrix Inequality, and Triangular Factorization for Linear Control Problems. J. Casti. (RM-76-010) $1.00 AS20.

Optimal Control Theory Problems with Network Constraints and Their Application. I. Zimin. (RM-76-013) $3.50 AS60.

Invariant Theory, the Riccati Group, and Linear Control Problems. J. Casti. (RM-76-059) $1.50 AS30.

Linking National Models of Food and Agriculture: An Introduction. M.A. Keyzer. (RM-77-002) $2.50 AS45.

Dual Systems of Dynamic Linear Programming. A.I. Propoi. (RR-77-009) $1.50 AS30.

An Algorithm for Constrained Optimization with Semismooth Functions. R. Mifflin. (RR-77-003) $2.50 AS45.

On the Distribution of the Hurst Range of Independent Normal Summands. A.A. Anis, E.H. Lloyd. (RR-77-016) $1.50 AS30.

A Practical Approach to Choosing Alternate Solutions to Complex Optimization Problems under Uncertainty. L.S. Belyaev. (RM-77-007) $2.50 AS45.

On the Theory of Max-Min. A.I. Propoi. (RM-77-013) $1.50 AS30.

The Dynamic Simplex-Method. A.I. Propoi, V.E. Krivonozhko. (RM-77-024) $3.50 AS60.

Please cite publication number when making an order. See inside back cover for order information.