A STUDY ON THE MODIFIED GRADIENT METHOD FOR QUASI-DIFFERENTIABLE PROGRAMMING

- 유사 미분가능 최적화 문제에 있어서 수정 급상승법에 대한 연구 -

김 준 홍*

요 지

변수의 어떤 값들에 대해 도함수를 가질 수 없는 함수를 최적화해야 하는 등, OR 에서는 여러 상황이 존재한다. 이것은 Convex Analysis[12]에서 이론적인 differential calculus 를 근저로하는 Non-differentiable Optimization 또는 Non-smooth Optimization 을 취급하는 것이 된다. 이러한 종류의 미분이 가능하지 않은 최적화문제는 연속함수를 위한 종래의 최적화법으로는 그 해법자체가 갖고 있는 연속성의 한계를 극복할 수 없다. 따라서, 이러한 문제를 해결하기 위해 Demyanov[4]가 제시한 quasi-differental function의 정의와 이들 함수에 따른 몇가지 주요정리들을 언급하고, 그것들을 토대로 Non-differentiable optimization problem의 수치적인 방법을 수행하기 위해 일종의 modified gradient 법을 제시한다. 이를 이용해서 numerical experiment를 위한 방법을 구체화하여, unrestricted non-differentable optimization problem에 적용하여, 그 수치해 결과를 보여서 그 타당성을 검토하였다.

1. INTRODUCTION

Consider the following optimization problem.

Min $f(x)$, s.t. $h_i(x) \leq 0$ for $i=1, \cdots, m$.,

where the objective function f and constraints $h_i(x)$ are real valued functions defined on R^n.

Regarding to classical theories for differentiable functions, a basis and well-known optimization method is gradient method. Gradient method generates a sequence of points $(x_k) \in S$, where $k \in N$, and S is feasible region for constraints, with *non-increasing* $\{f(x_k)\}, k \in N$, which shall converge to the required optimal solution. Usually the stationary point x_o is chosen from S, and at the k-th iteration the point x_k calculated by $x_k = x_{k-1} + \alpha_k g_{k-1}$, $0 < \alpha_k \in R$. The vector $g_k \in R^n$ is a direction in which the objective function value decreases, at least in a small neighourhood of x_k. For this, two subproblems, direction g_k and stepsize α_k, have to be solved. In the case that f, h_i are smooth, we have no problem to find a direction g_k and stepsize α_k. Many highly effective algorithms have been developed for those[1].

But unfortunately some problem cannot be represented as smooth, convex, and maximum functions. The function is subject to feasible set $S \subset R^n$ where the objective function and the set S do not have to be continuously differentiable. These problems could not be covered by the classical method, but are used to quasi-differentiable theory that has been introduced by V.F.Demyanov [4],[5], and it offers especially in the numerical point of view many advantages over Clarke's theory[2].

It provides *formulae involving equality relation* for many important operations, such as sum, product, quotient, maximum or minimum of finite number of functions while other calculi generally can insure only inclusions. It contains explicit definitions of how to calculate a steepest descent (ascent) direction. With a help of quasi-differentiable calculus, one cannot

---

* Dept. of Industrial Engineering, The University of SuWon, SuWon, Korea

접수 : 1992. 10. 17.
확정 : 1992. 10. 30.

only determine stationary points where the direction of optimization can be ignored, but one can distinguish the so called inf-stationary points from the sub-stationary points. The classical calculus for differentiable functions is included in the quasi- differentiable calculus.

In the following we will give a short survey of basic definitions and theorems.

2. CALCULUS FOR QUASI-DIFFERENTIALS

The calculation of quasi-differentiable functions is described to its full extent in [5]. In the following a brief description of this calculus is given, and it will be needed for subsequent statements and arguments.

Definition 2.1 A function f defined on an open set $S\in\mathbb{R}^n$ is said to be *quasi-differentiable* at a given point $x_o\in S$, if it is directionally differentiable in any arbitrary direction $g\in\mathbb{R}^n$ at x_o, and if there exist two convex, compacts set $\underline{\partial}f(x_o), \overline{\partial}f(x_o)\in\mathbb{R}^n$ such that the directional dervative is given by:

$$\frac{\partial}{\partial g} f(x_o) = \lim_{\alpha\to+0} \frac{1}{\alpha} f(x_o+\alpha g)-f(x_o)$$
$$= \max_{v\in\underline{\partial}f(x_o)} \langle v, g\rangle + \min_{w\in\overline{\partial}f(x_o)} \langle w, g\rangle \ , \ \forall g\in\mathbb{R}^n, \ \|g\|=1.$$

The pair $Df(x_o) = [\underline{\partial}f(x_o), \overline{\partial}f(x_o)]$ is called a *quasi-differential* of f at the point x_o, and the two sets $\underline{\partial}f(x_o), \overline{\partial}f(x_o)$ are called a *sub- and super-differential*, respectively.

Remark The quasi-differential is not unique, becaue if A is an arbitrary convex, compact set, then $[\underline{\partial}f(x_o)+A, \overline{\partial}f(x_o)-A]$ is also a quasi-differential for f at x_o .

Definition 2.2 A real-valued function f defined on an open set S is said to be *uniformly quasi-differential* at a given point $x_o\in S$, if it is quasi-differentiable and uniformly directionally differentiable at x_o.

The following theorem does not only state that the class of quasi-differentable functions is a linear space and closed with respect to taking pointwise maximum and minimum, but also specifies the quasi-differentiable obtained by these operations. It is therefore of major significance to the theory of quasi-differentiable functions.

Theorem 2.1 Let the function $f_i: \mathbb{R}^n\supset S\to \mathbb{R}$ be quasi-differentiable at $x_o\in S$, then the following statements hold:

1. $f(x) = \sum_{i=1}^{n}\lambda_i f_i(x)$, $\lambda_i\in\mathbb{R}$ is quasi-differentiable at x_o and $Df(x_o)$ is given by
   $Df(x_o) = \sum_{i=1}^{n} \lambda_i Df_i(x_o)$.
2. $f(x) = f_1(x)\cdot f_2(x)$ is quasi-differentiable at x_o and $Df(x_o)$ is given by
   $Df(x_o) = f_1(x_o)\cdot Df_2(x_o) + f_2(x_o)\cdot Df_1(x_o)$.
3. $f(x) = \frac{f_1(x)}{f_2(x)}$, $f_2(x_o)\neq 0$ is quasi-differentiable at x_o and $Df(x_o)$ is given by
   $Df(x_o) = \frac{f_1(x_o)\cdot Df_2(x_o) + f_2(x_o)\cdot Df_1(x_o)}{(f_2(x_o))^2}$.
4. $f(x) = \max_{i\in N} \{f_i(x)\}$ is quasi-differential at x_o and $Df(x_o)$ is given by
   $Df(x_o) = [\underline{\partial}f(x_o), \overline{\partial}f(x_o)]$, where $\underline{\partial}f(x_o) = \operatorname{co}_{k\in p}\{ \underline{\partial}f(x_o) - \sum_{i\in p,\, i\neq k} \overline{\partial}f_i(x_o) \}$,
   $\overline{\partial}f(x_o) = \sum_{k\neq p} \overline{\partial}f_i(x_o)$ for $P= \{j\in N \mid f(x_o)= f_j(x_o)\}$.

5. $f(x) = \min_{i\in N} \{f_{i(x)}\}$ *is quasi-differentiable at* x *and* $Df(x_o)$ *is given by*
$Df(x_o) = [\ \underline{\partial}f(x_o),\ \bar{\partial}f(x_o)]$, where $\underline{\partial}f(x_o) = \sum_{k\in p} \underline{\partial}f_k(x_o)$,
$\bar{\partial}f(x_o) = co\ \{\ \bar{\partial}f(x_o) + \sum_{k\in p}\ \sum_{i\in p,\ i\neq k} \underline{\partial}f(x_o)\ \}$ and $P = \{j\in N | f(x_o) = f(x_o)\}$.

Lemma 2.2 Let $f:\mathbb{R}^n\to\mathbb{R}$ be a quasi-differentiable function $x_o\in S$, S open. Then the steepest descent direction g^* of f at x_o is given by:

$$g^*(x_o) = - \frac{v_o+w_o}{\|v_o+w_o\|}, \quad \text{where} \quad \|v_o+w_o\| = \max_{w\in\bar{\partial}f(x_o)} \{ \min_{v\in\underline{\partial}f(x_o)} \|v+w\| \}$$

The vector, v_o-w_o, is the analogy of the classical steepest descent direction. In contrast to continuously differentiable functions, however, the direction of steepest descent of a quasi-differentiable function is not necessarily unique.

proof: The proof is straightforward and is omitted here.

Lemma 2.3 Let f be quasi-differentiable at $x_o\in S\subseteq\mathbb{R}^n$. For x_o to be the argument of a local minimum of f it is necessary that

$$-\bar{\partial}f(x_o) \subset \underline{\partial}f(x_o) .$$

Remark Point $x_o\in S$ satisfying Lemma 2.3 are said to be *inf-stationary*.

A sufficient condition for x_o to be the argument of a strict local minimum of a uniformly quasi-differenriable function f is given by

Lemma 2.4 Let f be uniformly quasi-differentiable at $x_o\in S\subseteq\mathbb{R}^n$. For x_o to be the argument of a local minimum of f it is sufficient that

$$-\bar{\partial}f(x_o) \subset \mathrm{int}(\underline{\partial}f(x_o)) .$$

Proof for Lemma 2.2 and 2.3 are given in Threorem 16.2 amd Threorem 16.7 in [5].

Based on these propositions and theories, we can now make an approach to optimization problems which are not covered by the classical classes of differentiable, convex or maximum functions.

3 A MODIFIED GRADIENT METHOD FOR QUASI-DIFFERENTIABLE FUNCTION

Assume that a factory where m units of products $i, i=1,2,\cdots,n.$, have to be distributed by a crane. Each product i should be carried to a place x_i. The crane, however, can transport the products in the plane's x- or y-direction only. In addition, it can carry more than one at a time. To determine the intitial position from where the distribution starts, the following model may be used :

Minimize the function f: $\mathbb{R}^2\to\mathbb{R}$, where $f(x) = \sum_{i=1}^{n} m_i|x_i-x|$, $m_i\in N$, $x, x_i\in\mathbb{R}^2$.

Obviously, this function is quasi-differentiable and it can be defined as follows:

min $f(x)$, $x\in\mathbb{R}^n$, where f: $\mathbb{R}^n\to\mathbb{R}$ is quasi-differentiable at every point $x\in\mathbb{R}^n$.

This problem is called an *unrestricted quasi-differentiable optimization problem*. The following problem is called the *restricted quasi-differentiable optimization problem*:

min $f(x)$, $x\in\mathbb{R}^n$, s.t. $h_i(x)\leq 0$, $i=1,\cdots,m.$, where the function $f:\mathbb{R}^n\to\mathbb{R}$ and $h_i(x):\mathbb{R}^n\to\mathbb{R}$ are quasi-differentiable functions.

The restricted quasi-differentiable optimization problem is less complicated than expected. It can be shown that the feasible set $S=\{x\in\mathbb{R}^n | h_i(x)\leq 0, i=1,\cdots,m.\}$ can be rewritten by $S=\{x\in\mathbb{R}^n | \max\{h_i(x)\}\leq 0, i=1,\cdots,m.\}$. The restricted quasi-differentiable optimization problem can be reformulated as follows:

Min $f(x)$, $x\in\mathbb{R}^n$, where $S = \{x\in\mathbb{R}^n \mid \max\{h_i(x)\}\le 0,\ i=1,\cdots,m.\}$.

This seems to be very convenient because the restricted problem can be reduced to a problem with only one constraint. This means that the problem with many restrictions is as difficult as the problem with one restriction. By means of one quasi-differentiable function, this method also allows the representation of feasible regions which are not necessary convex.

Theorem 3.1 Let function f, H, $h_i(x)$, $i=1,\cdots,n.$, be quasi-differentiable on $\mathbb{R}^n$ and $S= \{x\in\mathbb{R}^n \mid \max\{h_i(x)\}\le 0,\ i=1,\cdots,m.\} = H(x)\le 0$. Let $x_o\in S$, $H(x_o)=0$.
Calculate the quasi-differential of the function $\Psi(x)= \max\{f(x)-f(x_o),\ H(x)\}$, where

$$\underline{\partial}\Psi(x_o) = \mathrm{co}\{\underline{\partial}f(x_o)-\overline{\partial}H(x_o),\ \underline{\partial}H(x_o)-\overline{\partial}f(x_o)\},\ \overline{\partial}\Psi(x_o)=\overline{\partial}f(x_o)+\overline{\partial}H(x_o).$$

It is necessary for a point to be a minimum point of f on S hat $-\overline{\partial}\Psi(x_o)\subset\underline{\partial}\Psi(x_o)$.
Proof: see[2].

If this condition is not satisfied we can get a feasible descent direction $\overline{g}$ of f by taking a steepest descent direction of Ψ as the direction $\overline{g}$. To solve this kind of problem basing on the theoretical concept of previous section the two parts for the algorithm are recommended that the one is a criterion to decide whether a given point $x\in\mathbb{R}^n$ is optimal and the other is which have to be done, if a optimal solution is not reached. The logic answer to this subproblems is to choose a direction in which the function is decreasing campared with the actual point. For this we have to solve to two sub-problems: (1)How do we get such a direction? (2)How long do we have to progress in that direction until we check again whether another point is optimal?

This question is not a typical non-smooth problem, but for practical reasons we many have to modify the classical line search method, because it usually results in a one-dimensonal non-smooth optimization problem. So we have seen that quasi-differentiable theory can be used to answer these questions in a constructive search. The following generalized gradient method for unrestricted quasi-differentiable optimization problems is proposed.

Algorithm

Given the problem: min $f(x)$, $x\in\mathbb{R}^n$.

1. $k:=0$, x_k arbitrary.
2. Compute a quasi-differential $Df(x_k) = [\ \underline{\partial}f(x_k),\ \overline{\partial}f(x_k)]$.
3. Check whether the necesary condition for a local minimum $-\overline{\partial}f(x_k)\subseteq\underline{\partial}f(x_k)$ is satistied. If satisfied, go to 6.
4. Compute steepest descent direction $g^{(k)}(x_k) = \dfrac{v_o+w_o}{\|v_o+w_o\|}$ with
$$\|v_o+w_o\| = \max_{w\in\overline{\partial}f(x_o)}\{\min_{v\in\underline{\partial}f(x_o)}\|v+w\|\}.$$
5. compute an $\alpha^*\in\mathbb{R}$, where $\alpha^* = \min f(x_k+\alpha\cdot g_k(x_k))$, $\forall\alpha\in\mathbb{R}$.
$x_{k+1} := x_k+\alpha\cdot g_k(x_k)$, $k:= k+1$. go to 2.
6. $x^* := x_k$. stop.

Since the real number cannot be precisely represented on the computer, ε-technique has to be used for numerical calculation[9]. The setting up of an algorithm may require a lot of time. For this reason, a sophisticated and efficient implementation of the algorithm of the algorithm is very important for practical application.

For an algorithm to solve the quasi-optimization problem the steepest descent direction, a direction g_k in which the objective function value decreases, is to be determined.

At first, determine the two sets $\underline{\partial}f(x_o), \overline{\partial}f(x_o)$, then we get a steepest decent direction

by $g(x_o) = \dfrac{v_o+w_o}{\|v_o+v_o\|}$, $v_o, w_o \in \mathbb{R}^n$, $v_o+w_o \neq 0$

where $\|v_o+w_o\| = \max_{w\in\partial f(x_o)} \{ \min_{v\in\underline{\partial} f(x_o)} \|v+w\| \}$. ·····(*)

This can be reformulated as follows: $g(x_o) = -\dfrac{v_o-w_o}{\|v_o-w_o\|}$, $v_o, w_o \in \mathbb{R}^n$, $v_o-w_o \neq 0$.

where $\|v_o-w_o\| = \max_{w\in-\bar{\partial} f(x_o)} \{ \min_{v\in\underline{\partial} f(x_o)} \|v-w\| \}$. ·····(*)

Theorem 3.2 There exist an extremal point $w_o \in -\bar{\partial} f(x_o)$ and a $v_o(w_o) \in \underline{\partial} f(x_o)$, which satisfies equations (*).

Proof : Let w_o be an arbitrary element of $-\bar{\partial} f(x_o)$ satisfying(*). Take the set $P \subset \mathbb{R}^n$ of points x whose distance from $-\underline{\partial} f(x_o)$ does not exceed $\|v_o-w_o\|$, i.e.
$P := \{x\in\mathbb{R}^n \mid \|x-y\| \leq \|v_o-w_o\|, y\in\mathbb{R}^n\} = -\underline{\partial} f(x_o) + \|v_o-w_o\| \cdot B$, where B denotes the n-dimensional Euclidean ball. Let H be a supporting hyperplane to P containing w_o.

It is sufficient to show that at least one extremal point w_i of $-\bar{\partial} f(x_o)$ is an

element of $H \cap -\bar{\partial} f(x_o) \subseteq H \cap P$.

Since w_i are the extremal points, every point $x \in -\bar{\partial} f(x_o)$ can be represented by

$$x = \sum_{i=1}^{k} \lambda_i \cdot w_i \quad , \quad \sum_{i=1}^{k} \lambda_i = 1 \ , \ \lambda_i \geq 0.$$

Let $n \in \mathbb{R}^n$ be a vector which is normal to H and beyond $-\bar{\partial} f(x_o)$.

Then for every point $x \in H$, the following applies: $\langle n, w_o-x \rangle = 0$.

Assumtion: $w_i \notin H \cap -\bar{\partial} f(x_o) \Leftrightarrow w_i \in \{x\in\mathbb{R}^n \mid \langle n, w_o-x \rangle > 0\}$.

However, since $w_o \in -\bar{\partial} f(x_o)$, it follows $w_o = \sum_{i=1}^{k} \lambda_i \cdot w_i$, $\sum_{i=1}^{k} \lambda_i = 1$, $\lambda_i \geq 0$. $\Leftrightarrow w_o - \sum_{i=1}^{k} \lambda_i \cdot w_i = 0$.

Contradiction: $0 = \langle n, 0 \rangle = \langle n, w_o - \sum_{i=1}^{k} \lambda_i \cdot w_i \rangle = \sum_{i=1}^{k} \lambda_i \cdot \langle n, w_o - w_i \rangle > 0$.

Hence follows: $w_i \in H \cap -\bar{\partial} f(x_o)$. #

Since the set $-\bar{\partial} f(x_o)$ is given by a finite number of points $W := \{w_1, \cdots, w_m\}$, $w_i \in \mathrm{co}\{\mathrm{ext}(-\bar{\partial} f(x_o))\}$, where $\mathrm{ext}(-\bar{\partial} f(x_o))$ denotes the set of all extremal points of $-\bar{\partial} f(x_o)$, we can rewrite equations (*) as

$$g(x_o) = -\frac{v_o - w_o}{\| v_o - w_o \|} \ , v_o, w_o \in \mathbb{R}^n \ , \ v_o - w_o \neq 0,$$

$$\| v_o - w_o \| = \max_{w\in W} \{ \min_{v\in\underline{\partial} f(x_o)} \|v-w\| \} .$$

Besed on the proof in the Theorem 3.2, the problem is now reduced to finding the minimal distance between a point $w \in \mathbb{R}^n$ and a polyhedron $V := \mathrm{co}\{v_1, \cdots, v_j\}$. The difficulties obviously result from the fact that the polyhedron V is not given by a system of inequalities describing the intersection of affine halfspaces, but by a finite number of points which are not necessarily extremal points.

Let $h: V \rightarrow \mathbb{R}$, $h(v) = (\|w-v\|)^2$, $w \in \mathbb{R}^n$, $V \neq 0$, and

$h(v) = \left(\dfrac{dh(v)}{dv_1} , \cdots , \dfrac{dh(v)}{dv_n} \right)$: the gradient of h at v.

To minimize h, we may proceed as follows:

Start: Let $k := 0$, $V_o := \{v_1, \cdots, v_j\}$, $v_k \in V_o$ arbitrary.

Iteration 1. Find the set of all points $\bar{V} \subset v_k$ such that $\{w\}$ and $\bar{V}$ are subsets of the same affine halfspace determined by the hyperplane $H_k := \{x\in\mathbb{R}^n \mid \langle \nabla h(v_k), x-x_k \rangle = 0\}$ i.e. $\bar{V} := \{x\in V_k, \ x \neq v_k \mid \mathrm{sign}(\langle \nabla h(v_k), w-v_k \rangle) = \mathrm{sign}(\langle \nabla h(v_k), x-v_k \rangle)\}$.

If $V=\bar{\Phi}$, v_k minimizes h.

Iteration 2. Minimize the function $h_k: V \to \mathbb{R}$, where $v \to \|x-v\|^2$, $x \in H^k$.
Since the distance between a point $x \in \mathbb{R}^n$ and a hyperplane $H=\{y \in \mathbb{R}^n \mid f(y)=\alpha\}$ is given by: $\text{dist}(x,H) = \frac{|f(x)-\alpha|}{\|f\|}$.

This can be calculated by: minimize $\frac{|\langle \nabla h(v_k), v\rangle - \langle \nabla h(v_k), v_k\rangle|}{\|\nabla h(v_k)\|}$.

$\nabla h(v_k)=0$ indicate that $v_k=w$ is a minimizer of h. Thus, suppose that $\nabla h(v_k) \neq 0$.
Now since $\|\nabla h(v_k)\|$=constant for all $v \in \bar{V}$, it is sufficient to calculate
$v^* = \arg\min_{v \in V} (|\langle \nabla h(v_k), v-v_k\rangle|)$.

Iteration 3. Minimize the function $T: [0,1] \to \mathbb{R}$, where $t \to \|w-t \cdot v_k-(1-t) \cdot v^*\|$. Using classical calculus, we obtain the minimum for
$t^* = \min (1, \max (0, \frac{\langle w-v_k, v_k-v^* \rangle}{\|v^*-v_k\|}))$.

Iteration 4. $v_{k+1} := t^* \cdot v_k+(1-t) \cdot v^*$, $v_{k+1} := v_0 \cup \{v_{k+1}\}$, k=k+1, and perform the next iteration.
Stopping rule: Consider $w \subseteq V \Leftrightarrow \|v_{k+1}-v_k\| < \varepsilon$.

Remark: Numerical experience has shown, that this algorithm can be improved by modifying step 2. i.e. be replacing $v^* = \arg\min_{v \in V} (|\langle \nabla h(v_k), v-v_k\rangle|)$

with $v^* = \arg\min_{v \in V} \frac{|\langle \nabla h(v_k), x-x_k\rangle|}{\|v-v_k\|}$

4 IMPLEMENTATION

To implement an algorithm solving optimization problems by the method described in the previous section, we will consider to three basic phases: (a) calculation of the funtion value and a quasi-differential, (b) finding a steepest descent direction, and (c)solving the line search problem.

4.1 Calculation of the function value and a quasi-differential

To calculate a function value, it needs to a method for recognize, analyse and evaluate the function describing the optimization problem. Because the quasi-differentiable functions do not have a fixed structure in contrast to linear functions where the these functions are represented by its coefficients, the structure is not known run time of the program in advance. Thus, by using a special technique from the field of compiler construction which is the interpretation of arithmetic expression, we use a method to recognize the function during run time of the program. Language implementation depends always on language definition, because a compiler has to know which structure is permitted and which is prohibited. The syntax of a language determines which character strings constitute well formed programs in the language and which do not. The syntactic rules of a lanuage can be assigned to different levels according to their meaning. The lowest level contains the spelling rules for basic symbols. They describe the strucure of logical units which cannot be further divided, such as keywords, identifiers or special symbols. This lowest level is handled by lexical analysis or scanning. The second level of syntactical rules is called *concrete syntax*. Concrete syntax rules describe the assembling of language structures such as statements or expressions which are made up of basic symbols, i.e. they split up the structure into units which can be recognized by scanner. This process is called *structural analysis* or *parsing*.

For the more detail survey of this technique is described complehensively by Davie and Morrison[3].

4.1.1 Evaluation

The parsing process provides an evaluation tree. An inner node is always marked by an operator and a leaf is always indicated an operand[3]. The tree is stored in a chained list, this is, each node is represented by an array which contains some information which is name of operator, value of operand, type of node, and, at the same time, two elements which provide the addresses of the nodes to the left and right. To calculate the value of an arithmetic express from this tree, we use a cellar technique.

Definition 4.2.1 *A cellar* can be described as an area of storage where the data can be entered only at top. Thus, data already stored there, will be pushed down. Accordingly one can only reference or change the top elements. When no longer needed, the top elements are deleted, thus popping up the elements below. This is simply an implementation of the principle *Last-In-First-Out*.

The usual method of implementing a cellar is to use an array S and a counter i. If i=n where n>0, then the cellar contains $S[1],\cdots,S[n]$ is the top elements. For calculation purposes, we now travel the evaluation tree such that we get successive a Postfix notation in the cellar. If we get an operand from the tree, we just move it to the cellar. If we get an operator, then the two operends on top of the cellar will be evaluated, the two operands will be deleted and the result is placed on top of the cellar. Having traveled through the whole tree, the cellar is reduced to one element which now contains the final result.

4.2 Computation of a steepest direction

As mentioned in section 2, two sets can be calculated by taking the differentiable parts of the function as primitives, and constructing the higher parts with help of these rules to calulate the quasi-differential function. Consider a complehensive example as follows:

$$f: \mathbb{R}^2 \rightarrow \mathbb{R}, \text{ where } (x_1, x_2) \rightarrow \max(x_1 + x_2, \min(x_1, x_2)).$$

The first thing we have to do is divide the formular into several simpler parts in order to get the differential parts of the expression. We can calculate the quasi-differential at the point $x_o = (0,\cdots,0)$.

We define:
$$f(x_1,x_2) = \max(h_1(x_1,x_2), h_2(x_1,x_2))$$
$$h_1(x_1,x_2) = f_1(x_1,x_2)+f_2(x_1,x_2))$$
$$h_2(x_1,x_2) = \min(f_1(x_1,x_2), f_2(x_1,x_2))$$
$$f_1(x_1,x_2) = x_1$$
$$f_2(x_1,x_2) = x_2.$$

In order to simulate this problem on a computer, we need a criterion of how to divide an arbitrary expression analogously, that means we need a method of defining the functions $h_1,\cdots,h_k$, $f_1,\cdots,f_j$ automatically. When looking at the evaluation tree shown below it becomes obvious that these functions correspond to the nodes of the tree. Thus, this problem is not a real problem since we automatically get the correct divisions when we travel the tree and call the procedures for the calculation rules in accordance with the actual node. Theoretically, this proceeding can be continued until there are only the variables $x_1, x_2, \cdots, x_k$ left on the lowest level of the tree. The program will be then calculate the quasi-differential parts, i.e. the lowest level in our program consists of the differential parts of the function. As above example, we achive the correct division into differentiable parts by parsing the function f. Considering the two rules in section 4.1, this is exactly what we need in order to calculate a quasi-differential. In fact, this is the very simple solution for the problem of calculating quasi-differentials. The only difference between calculating the function value and calculating

the quasi-differential is that, there are not only real number in the case of quasi-differentials but matrices or convex hulls of vectors. Thus, it needs two cellars instead of one, namely one cellar for the sub-differential and one for the super-differential. Furthermore, the new procedures for addition, substraction and scalar multiplication of convex hulls are to be implemented.

Since we do not know an algorithm which is fast enough to determine the convex hull of a set of points, we do not eliminate inner points of a convex hull, i.e. we store all points except identical ones which we get during the calculation process. But this does not cause any problems, since we only need a kind of modified distance between convex hulls to determine the steepest descent.

4.3 Line search method for quasi-differentiable programming

Based on the above determination of direction g_k in quasi-differentiable optimization problem, stepsize is to be determined by a given point x_k and a given direction g_k. This problem is usually called the *line search problem*. To be more specific, let us solve the one dimensional optimization problem:

$$\text{Min } h_k: U \in R \to R \text{ , where } t \to f(x_k + t \cdot g_k).$$

This can be achived by constructing a sequence $(t_i), i \in N$, which has to converge to the minimum of function h_k. The optimal solution t^* is the required value α_k. Then, two subproblems, recursion rule and stoping rule, are suggested. The algorithm is supposed to approach the neighbourhood of the optimum with a minimal number of steps. The is particular important when a large distance has to he covered, since compulational time is limited. Thus, a sufficiently large step size must be selected. Selecting of a very small step size in the neighbourhood of the optimum in order to get as much exact result as possible.

We can now test the function h_k inside the interval by means of several methods, and we do not have to consider other points outside. One possibility to examine the function h_k inside the interval may be approximate the function by simpler functions. These can be calculated much faster and we can get the minimum by one calculation. Sequential search methods are widely used for differentiable functions. They can also be applied to quasi-differentiable functions for the following reasons: Because these search techniques are not analytical, we get a finite number of function values only. This also means that we do not need any derivatives but just compare function values and then decide how to shrink the interval by elimlnating portions of the current interval.

Sequential search methods for the quasi-differentiable case require the unimodal property, that is, the interval is presumed to include one and only one point at which the function has a global minimum for this interval to guarantee the approximation of a global optimum. Obviously this property is not generally fulfilled for the quasi-differentiable case, but since we do not search for global optima, we may ignore this property for our purposes. We will provide a description of specific sequential searchs, Fibonacci search.

The Fibonacci sequence $(F_n), n \in N$, may be used for searching on intervals. The Fibonacci search is initialized by determining the smallest Fibonacci number $F_N \in (F_n), n \in N$, that satisfies: $F_N \cdot \varepsilon \geq t_1 - t_o$, where ε is a prescribed tolence and $[t_o, t_1]$ is starting interval. $\bar{\varepsilon} := \dfrac{t_1 - t_o}{F_N}$

The first two points in the search are located within the interval $[t_o, t_1] F_{N-1} \cdot \varepsilon$ ūnits from the endpoints of the interval. Take the interval again formed by the two subintervals to the left and right of the point of the minimal objective value as new current interval.

Successive points are positioned within the current interval $F_j \cdot \varepsilon (j = N-2, \cdots, 2)$ units from the

latest endpoint. Note that with the Fibonacci search procedure we can state in advance the number of function evaluations that will be required to achive a certain accuracy. Moreover, that number is independent from the structure of the particular quasi- differentiable function.

5 COMPUTATIONAL OBSERVATIONS AND RESULTS

The following example will be briefly demonstrate the behavior of our algorithm.
Given the unrestricted optimization problem:

$$f : \mathbb{R}^9 \to \mathbb{R}, \text{ where } f(x) = \sum_{i=1}^{9} |x_i|, \quad x=(x_1, \cdots, x_9).$$

The following is succesive iteration values by computational results.

i	x_i	$f(x_i)$	g_i	α_i
0	(132, 45, -33, 7, 0, 90, 88, -21, -900)	1316	0.3535(-1, -1, 1, -1, 0, -1, -1, 1, 1)	127
1	(87, 0, 12, -38, 0, 0, 45, 43, 24, -855)	1104	0.3779(-1, 0, -1, 1, 0, -1, -1, -1, 1)	114
2	(44, 0, -31, 5, 0, 2, 0, -19, -812,)	913	0.4082(-1, 0, 1, -1, 0, -1, 0, 1, 1)	64.5
3	(17.6, 0, -4.6, -21.4, 0, 24.4, 0, 7.4, -785.6)	861	0.4082(-1, 0, 1, 1, 0, 1, 0, -1, 1)	43.4
4	(-0.1, 0, 13.1, -3.6, 0, -6.6, 0, -10.4, -767.9)	802	0.4082(1, 0, -1, 1, 0, 1, 0, 1, 1)	24.8
5	(10.0, 0, 3.0, 6.5, 0, 3.5, 0, -0.25, -757.8)	781	0.4082(-1, 0, -1, -1, 0, -1, 0, 1, 1)	8.6
6	(6.5, 0, 0.5, 3.0, 0, 0, 0, 3.25, -754.3)	767.6	0.4472(-1, 0, 1, -1, 0, 0, 0, -1, 1)	7.25
7	(3.3, 0, 2.7, -0.25, 0, 0, 0, 0, -751.0)	757.3	0.5(-1, 0, -1, 1, 0, 0, 0, 0, 1)	5.5
8	(0.5, 0, 0, 2.5, 0, 0, 0, 0, -758.6)	751.3	0.5(-1, 0, 1, -1, 0, 0, 0, 0, 1)	1.1
9	(0, 0, 0.5, 2.0, 0, 0, 0, 0, 0, -747.75)	750	0.5773(0, 0, -1, -1, 0, 0, 0, 0, 1)	3.35
10	(0, 0, -1.4, 0, 0, 0, 0, 0, 0, -745.8)	747	0.7071(0, 0, 1, 0, 0, 0, 0, 0, 1)	1.9
11	(0, 0, 0, 0, 0, 0, 0, 0, 0, -744.4)	744.4	(0, 0, 0, 0, 0, 0, 0, 0, 1)	744.4
12	(0, 0, 0, 0, 0, 0, 0, 0, 0)			

We think that this method seems very promising for first order algothms. It may be extended to include some other problems such as global optimization or vector optimization for the quasi-differentiable case. For higher order type algorithms, however, you should better use other methods which do not require sets to calculate a descent. The algorithm seems to be the first which has been implemented for the general case of quasi-differentiable optimization. Thus, it is only a matter of course that it is very convenient for simpler problems with a special structure, linear quadratic or convex optimization problem, because of its time and space requirements.

The implementation was performed by means of the programming language FORTRAN 77. The numerical results was supposed to provide only an experimental code for treating quasi-differentiable optimization problems. For commercial purposes, it has to be elaborated therefore in order to speed up execution time and reduce main memory. The parameters of the code are designed for problems up to dimension 16 and for 250 constraint functions. Then the parameters will be able to cope with problem up to dimension of 50 and a number of 500 for the constraints. But since this code has been specifically developed as an experimental code for scientific purposes, further research in this field may considerably speed up the algrithm.

LITERATURS

1. Bazaraa, M.S., Shetty, C.M. "Nonlinear Programming", John Wiley and Sons, 1979.
2. Clarke, F.H., "Generalized gradients and applications", Trans. Amer. Math. Soc., Vol.205, 1975.
3. Davie, A.J.Y.T., Morrison, R., " Recursive descent compiling", Halsted Press, New York, 1978.
4. Demyanov, V.F., Rubinov, A.M." On quasi-differentiable mappings", Optimization Vol.14, No.1, 1983.
5. Demyanov, V.F., Rubinov, A.M., " Quasi-differentiable calculus", Optimization Software, Springer-Verlag, Berlin-Heidelberg-New York-Tokyo, 1986.
6. Demyanov, V.F., Vasilev, L.V., " Non-differentiable optimization", Optimization Software, Springer-Verlag Berlin, 1985.
7. Demyanov, V.F., A.M.Rubinov, "On quasi-differentiable functionals", Soviet Mathematics Doklady, Vol.21, P.14-17, 1980.
8. Demyanov, V.F., L.N.Polyakova, "Minimization of a quasi-differentiable funtion on a quasi-differential set", USSR Computational Mathematics and Mathematical Physics, Vol.20, No.4, P.34-43, 1981.
9. Pallaschke, D., P.Recht, "On the steepest descent method for a class of quasi-differentiable optimization problems" Nondifferentiable Optimization: Motivations and Applications Preceedings, Sopron, Hungary, 1984, pp.252-263, Springer-Verlag Brelin, Heidelberg, New York, Tokyo 1985.
10. Poljak, B.T., "Subgradient methods: A survey of soviet research", Nonsmooth optimization, Pergamon Press, Oxford, 1978.
11. Polyakova, L.N., " necessary condition for an extremum of quasi-differentiable functions", Vestnik Leningrad University Mathematics, Vol.13, P.241-242, 1981.
12. Rockafellar, R.T., " convex analysis", Princeton Unversity Press, Princeton, 1979.