Abstract

The paper presents a new descent algorithm for locally Lipschitz continuous functions $f : X \to \mathbb{R}$. The selection of a descent direction at some iteration point $x$ combines an approximation of the set-valued gradient of $f$ on a suitable neighborhood of $x$ (recently introduced by Mankau & Schuricht [22]) with an Armijo type step control. The algorithm is analytically justified and it is shown that accumulation points of iteration points are critical points of $f$. Finally the algorithm is tested for numerous benchmark problems and the results are compared with simulations found in the literature.

1 Introduction

In this paper we present a new descent algorithm to find local minima or critical points of a locally Lipschitz continuous function $f : X \to \mathbb{R}$ on a Hilbert space $X$. For the minimization of a nonsmooth function

$$f(x) \to \min \ (x \in X)$$  \hfill (1.1)

numerous algorithms based on quite different methods have been proposed in the literature. Let us mention, without being complete, bundle-type methods (cf. Alt [2], Frangioni [10], Gaudioso & Monaco [11], Hiriart-Urruty [10], Kiwiel [17], Makela & Neittaanmaki [21], Mifflin [23], Schramm [27], Wolfe [31], Zowe [33]), proximal point and splitting methods as e.g. the Fista or the primal dual method (cf. Beck [4], Eckstein & Bertsekas [9], Chambolle & Pock [6]), gradient sampling algorithms (cf. Burke, Lewis & Overton [5], Kiwiel [18]), algorithms based on smoothing techniques (cf. Polak & Royset [25]) and the discrete gradient method (cf. Bagirov & Karasozen [3]).

Bundle-type methods, proximal point methods, and splitting methods require $f$ to be convex or to have some other special structure. Many algorithms for locally Lipschitz continuous functions as the discrete gradient method need to know the entire generalized gradient of $f$ at given points. Stochastic methods like the gradient sampling algorithm are robust without the knowledge of the entire generalized gradient, but at the cost of high computational effort. Therefore they are limited to minimization problems on low dimensional spaces.

Recall that the derivative $f'(x)$ indicates a direction of descent for $f$ near $x$. However if the direction of descent changes rapidly in a small neighborhood of $x$, which is typical for functions $f$ having large second derivatives or that are even nonsmooth, then some knowledge of $f$ on a whole neighborhood of $x$ is necessary for the determination of a suitable direction of descent near $x$.

For a new robust and fast algorithm we combine ideas of bundle-methods and gradient sampling methods. We use the concept of gradients of $f$ on sets as introduced in Mankau & Schuricht [22] (which extends ideas from Goldstein [14]). Here, similar to gradient sampling methods, generalized gradients of $f$ on a whole neighborhood of $x$ are considered for the determination of a suitable descent direction near point $x$. But, in contrast to e.g. Burke,
Lewis & Overton [5] and Kiwiel [18], the set-valued gradient on a neighborhood of point \( x \) is not approximated stochastically. We rather use an elaborate recursive inner approximation coupled with the computation of related descent directions until a generalized Armijo condition is satisfied (a condition similar to that used in Alt [2] and Schramm [27] in connection with the \( \varepsilon \)-subdifferential). Finally a line search along a direction of sufficient descent gives the next iteration point (cf. Pytlak [26]). For better performance we may also adapt the norm of \( X \) in each step. It turns out that our algorithm requires substantially less gradient computations than in [5] and [18]. Therefore it is also applicable on high dimensional spaces as needed for variational problems.

For a locally Lipschitz continuous function \( f : X \to \mathbb{R} \) our methods merely demand that, at any point \( x \), both the value \( f(x) \) and at least one element of the generalized gradient \( \partial f(x) \) (in the sense of Clarke [8]) can be computed. Notice that this mild requirement is assumed in any of the above mentioned gradient based algorithms and that it is typically met in applications. In an upcoming paper an extended algorithm is presented where quasi-Newton methods and preconditioning methods are included by a suitable change of norm in each iteration step.

Section 2 gives a brief overview about gradients on sets as needed for our treatment. The algorithm and some convergence results are given in Section 3. After the formulation of the condition of sufficient descent and of several general assumptions, Section 3.1 provides the main Algorithm 3.8 and its properties. Algorithm 3.8 calls Algorithm 3.14 for the computation of a suitable inner approximation of the set-valued gradient on a neighborhood of the current iteration point and the computation of a related descent direction while Step 3 of Algorithm 3.14 calls Algorithm 3.17 for some subiteration. Figure 1 gives an overview of the whole algorithm and several statements justify essential steps of it. Theorem 3.24 shows that every accumulation point of iteration points produced by Algorithm 3.8 is a critical point in the sense of Clarke. The proofs are collected in Section 3.2. Comprehensive numerical tests of our algorithm for classical benchmark problems can be found in Section 4. Here the simulations are also compared with results from Burke, Lewis & Overton [5], Kiwiel [18], Alt [2], Schramm [27] and the BFGS algorithm.

Notation: \( X \) is a Hilbert space with scalar product \( \langle \cdot, \cdot \rangle \) where the dual \( X^* \) is always identified with \( X \). For a set \( M \) we write \( \overline{M} \) for its closure, \( \text{conv} M \) for its convex hull and \( \text{conv}^c M \) for its closed convex hull. \( B_\varepsilon(x) \) and \( B_\varepsilon(M) \) are the open \( \varepsilon \)-neighborhood of point \( x \) and set \( M \), respectively. \( [x, y] \) stands for the open segment between \( x, y \) and, in particular, \( [x, y] \subset \mathbb{R} \) for the open interval. \( \mathbb{R}_{>0} \) denotes the positive real numbers. For a locally Lipschitz continuous function \( f : X \to \mathbb{R} \) we write \( f^0(x; h) \) for Clarke’s generalized directional derivative of \( f \) at \( x \) in direction \( h \) and \( \partial f(x) \subset X \) for Clarke’s generalized gradient of \( f \) at \( x \) (cf. Clarke [8]).

2 Gradient on sets

Let \( f : X \to \mathbb{R} \) be a locally Lipschitz continuous function on a Hilbert space \( X \). Clarke’s generalized gradient \( \partial f(x) \) of \( f \) at \( x \) and the corresponding generalized directional derivative \( f^0(x; h) \) of \( f \) at \( x \) in direction \( h \) somehow express the behavior of \( f \) at point \( x \) (cf. Clarke [8]). However, for the construction of a descent step in a numerical scheme, some information about the behavior of \( f \) on a whole neighborhood of \( x \) is useful in general. In particular, for describing the behavior of \( f \) on the whole \( \varepsilon \)-ball \( B_\varepsilon(x) \), we use some set-valued gradient \( \partial^s f(x) \) of \( f \) and some corresponding generalized directional derivative \( f^s(x; h) \) as introduced in Mankau & Schuricht [22] by using Clarke’s pointwise quantities. For the convenience of the reader we present some brief specialized summary of that material as needed for our treatment.

\footnote{Notice that any Hilbert space is uniformly convex and reflexive.}
For $\varepsilon > 0$ we define the gradient of $f$ on $\overline{B_\varepsilon(x)}$ by
\[
\partial f(x) := \text{conv} \bigcup_{y \in \overline{B_\varepsilon(x)}} \partial f(y) \tag{2.1}
\]
(notice that the closed convex hull $\text{conv}$ agrees with the weakly closed convex hull) and the directional derivative of $f$ on $\overline{B_\varepsilon(x)}$ in direction $h \in X$ by
\[
f^0_\varepsilon(x; h) := \sup_{y \in \overline{B_\varepsilon(x)}} f^0(y; h). \tag{2.2}
\]

We have the following basic properties (cf. Proposition 2.3 and Corollary 2.10 in [22]).

**Proposition 2.3.** Let $f : X \to \mathbb{R}$ be Lipschitz continuous of rank $L$ on a neighborhood of $\overline{B_\varepsilon(x)}$ with $x \in X$ and $\varepsilon > 0$. Then

1. $\partial f(x)$ is nonempty, convex, weak-compact and bounded by $L$.
2. $f^0_\varepsilon(x; \cdot)$ is finite, positively homogeneous, subadditive, and Lipschitz continuous of rank $L$. Moreover it is the support function of $\partial f(x)$, i.e.
\[
f^0_\varepsilon(x; h) = \max_{a \in \partial f(x)} \langle a, h \rangle \text{ for all } h \in X. \tag{2.4}
\]
3. We have
\[
\partial f(x) = \{ a \in X \mid \langle a, h \rangle \leq f^0_\varepsilon(x; h) \text{ for all } h \in X \}. \tag{2.5}
\]
4. Let $h \in X$ with $f^0_\varepsilon(x; h) < 0$ and let $t > 0$ with $\|x + th\| \subseteq \overline{B_\varepsilon(x)}$. Then
\[
f(x + th) \leq f(x) + tf^0_\varepsilon(x; h) < f(x). \tag{2.6}
\]
5. Let $\varepsilon_k \to 0$ with $\varepsilon_k > 0$ and let $h \in X$. Then
\[
\lim_{k \to \infty} f^0_{\varepsilon_k}(x; h) = f^0(x; h) \quad \text{and} \quad \cap_{k \in \mathbb{N}} \partial f^k(x) = \partial f(x). \tag{2.7}
\]

Regularity of $f$ at $x$, i.e. $0 \notin \partial f(x)$, implies regularity of $f$ on some $\overline{B_\varepsilon(x)}$ by Proposition 2.16 in [22].

**Lemma 2.6.** Let $f : X \to \mathbb{R}$ be locally Lipschitz continuous and let $0 \notin \partial f(x)$ for some $x \in X$. Then there exist $\varepsilon > 0$ and $h \in X$ with $\|h\| = 1$ such that
\[
-\|a\| \leq \langle a, h \rangle \leq f^0_\varepsilon(x; h) < 0 \quad \text{for all } a \in \partial f(x). \tag{2.8}
\]

Moreover, $0 \notin \partial f(x)$ by (2.5).

Motivated by Proposition 2.3 (4) we say that $h \in X$ is a descent direction of $f$ on $\overline{B_\varepsilon(x)}$ if $f^0_\varepsilon(x; h) < 0$. We call $h$ steepest or optimal descent direction of $f$ on $\overline{B_\varepsilon(x)}$ (with respect to $\|\cdot\|$) if
\[
\|h\| = 1 \quad \text{and} \quad f^0_\varepsilon(x; h) = \min_{\|h\| \leq 1} f^0_\varepsilon(x; h) < 0. \tag{2.9}
\]

Theorem 3.10 of [22] ensures the existence of optimal descent directions and of norm-minimal elements in $\partial f(x)$. 

3
Proposition 2.8. Let \( f : X \to \mathbb{R} \) be Lipschitz continuous on a neighborhood of \( \overline{B}_\varepsilon(x) \) for some \( x \in X, \varepsilon > 0 \). Then there is a unique \( \tilde{a} \in \partial f(x) \) with

\[
\|\tilde{a}\| = \min_{a \in \partial f(x)} \|a\|.
\]

(2.9)

If \( 0 \notin \partial f(x) \) or, equivalently, \( f^0(x; h) < 0 \) for some \( h \in X \) (cf. (2.5)), then there is a unique optimal descent direction \( h \) on \( B_\varepsilon(x) \). In particular

\[
\tilde{h} = -\frac{\tilde{a}}{\|\tilde{a}\|}, \quad f^0_\varepsilon(x; \tilde{h}) = \min_{\|h\| \leq 1} f^0_\varepsilon(x; h) = -\|\tilde{a}\|.
\]

(2.10)

Corollary 3.15 and Corollary 3.16 in [22] state some stability of descent directions.

Corollary 2.11. Let \( f : X \to \mathbb{R} \) be Lipschitz continuous of rank \( L \) on a neighborhood of \( \overline{B}_\varepsilon(x) \) for some \( x \in X, \varepsilon > 0 \), let \( 0 \notin \partial f(x) \), and let \( \tilde{a}, \tilde{h} \) be as in Proposition 2.8. Then every \( h \in X \) with \( \|h - \tilde{h}\| < \frac{\|\tilde{a}\|}{L} \) is a descent direction on \( B_\varepsilon(x) \).

This allows to get descent directions by suitable approximations of \( \tilde{a} \), which is important for our numerical algorithms.

Corollary 2.12. Let \( f : X \to \mathbb{R} \) be Lipschitz continuous on a neighborhood of \( \overline{B}_\varepsilon(x) \) for some \( x \in X, \varepsilon > 0 \), let \( 0 \notin \partial f(x) \), and let \( \tilde{a}, \tilde{h} \) be as in Proposition 2.8. Then for any \( \delta \in ]0,1[ \) there is some \( \tau > 0 \) such that for every \( a' \in \partial f(x) \) with

\[
\|a'\| \leq \min_{a \in \partial f(x)} \|a\| + \tau = \|\tilde{a}\| + \tau
\]

we have that \( h' := \frac{a'}{\|a'\|} \) is a descent direction on \( B_\varepsilon(x) \) and satisfies

\[
f^0_\varepsilon(x; h') \overset{\text{2.1}}{=} \max_{a \in \partial f(x)} \langle a, h' \rangle < -\delta \|\tilde{a}\|.
\]

3 Descent algorithm

We now introduce some descent algorithm for locally Lipschitz continuous functions \( f : X \to \mathbb{R} \) on a Hilbert space \( X \). At each iteration point \( x \) we determine an approximation \( a \) of the norm-minimal element \( \tilde{a} \in \partial f(x) \) (cf. (2.9)) with respect to some suitable radius \( \varepsilon > 0 \). We are interested in pairs \((a, \varepsilon)\) satisfying a condition of sufficient descent in the sense of a generalized Armijo step of the form

\[
f(x - \varepsilon h) - f(x) \leq -\delta \varepsilon \|a\| \quad \text{with} \quad h = \frac{a}{\|a\|}
\]

(3.1)

where \( \delta \in ]0,1[ \) is fixed for the whole scheme. As new iteration point we then select \( x - \sigma h \) for some \( \sigma \leq \varepsilon \) such that (3.1) still holds with \( \sigma \) instead of \( \varepsilon \). If \( 0 \notin \partial f(x) \), the norm \( \|a\| \) will be very small and the null step condition

\[
\|a\| < T_\varepsilon(\varepsilon)
\]

(3.2)

(with a suitable control function \( T_\varepsilon \) that is fixed for the whole scheme) indicates that situation. Here we cannot expect (3.1) in general and we have two possibilities. If \( \varepsilon \) is on the desired level of accuracy for the minimizer (or critical point), we can stop the algorithm. Otherwise the used ball \( \overline{B}_\varepsilon(x) \) is too large for an iteration step with sufficient descent. Therefore we decrease \( \varepsilon \) and look for sufficient descent with a new pair \((a, \varepsilon)\). Our approximation of \( \tilde{a} \) combined
with the analytically justified step size control ensures that we always get sufficient descent for some \( \varepsilon > 0 \) small enough (cf. Lemma 2.4 and also the proof of Theorem 3.2.1). That we finally end up with the null step condition on the desired scale, \( \varepsilon \) has to become sufficiently small during the algorithm, which is ensured by control functions \( T_1 \) and \( T_2 \). But, that the algorithm doesn’t get stuck in a small ball without critical point, \( \varepsilon \) shouldn’t approach zero too fast, which is ensured by control functions \( T_1 \) and \( G \). Thus a careful selection of the step size, that is related to \( \varepsilon \), plays a very important role. The algorithm can be improved by choosing suitable equivalent norms at each iteration step.

Let us start with general requirements for the control functions \( T_1 \), \( T_2 \), and \( G \).

**Assumption 3.3.** Suppose that:

1. \( T_1, T_2 : \mathbb{R}_{>0} \to \mathbb{R}_{>0} \) are non-decreasing functions such that
   
   \[
   \lim_{t \to 0} T_1(t) = 0 \quad \text{and} \quad \lim_{n \to \infty} T_2^n(t) = 0 \quad \text{for all} \ t > 0
   \]
   
   where \( T_2^n \) is given inductively by \( T_2^1 = T_2 \) and \( T_2^n = T_2 \circ T_2^{n-1} \) for all \( n \in \mathbb{N} \). Notice that this implies
   
   \( T_2(t) < t \) \quad \text{for all} \ t > 0
   
   (otherwise \( T_2(t) \geq t \) for some \( t > 0 \) and, since \( T_2 \) is non-decreasing, induction would give \( T_2^n(t) \geq t^{\frac{n}{n-1}} 0 \)).

2. \( G : \mathbb{R}_{>0}^2 \to \mathbb{R}_{>0} \) is a function having at least one of the following properties:
   
   a) \( G(x_k, y_k) \to 0 \) implies \( x_k \to 0 \) for any sequences \( (x_k) \) and \( (y_k) \).

   b) For any \( x_0 > 0 \) there is some \( y_0 \) such that \( G(x, y) \geq y \) for all \( x > x_0 \) and \( y < y_0 \).

Since the conditions for \( T_2 \) and \( G \) are quite technical, we provide some typical examples.

**Example 3.4** (examples for \( T_2 \) and \( G \)).

1. \( T_2(x) := ax \) for \( a \in [0, 1[ \).

2. \( T_2(x) := \frac{x}{1+x} \) where, in particular, \( T_2\left(\frac{1}{n} \right) = \frac{1}{n+1} \).

3. \( G(x, y) \geq \alpha \) with a constant \( \alpha > 0 \) satisfies (a).

4. \( G(x, y) = \alpha y \) with \( \alpha \geq 1 \) satisfies (b).

5. \( G(x, y) = \overline{G(x)} \) with \( \overline{G} : \mathbb{R}_{>0} \to \mathbb{R}_{>0} \) non-decreasing and \( \lim_{t \to 0} \overline{G}(t) = 0 \) satisfies (a).

6. \( G := G_1 + G_2 \) and \( G := \max \{G_1, G_2\} \) satisfy (a) or (b) if \( G_1, G_2 \) satisfy both (a) or (b), respectively.

7. \( G(x, y) := \max \{\alpha, y\} \) satisfies (a) and \( G(x, y) := \min \{\alpha, y\} \) satisfies (b) for any \( \alpha > 0 \).

As already mentioned, it might be useful to adapt the norm in every iteration (recall that the Newton method can be considered as descent algorithm with changing norm at each step). In our algorithm we allow a change of norm in every step as long as we have some uniform equivalence.

**Assumption 3.5.** The norms \( \|\cdot\|_k \) and \( \|\cdot\| \) on \( X \) are uniformly equivalent, i.e. there is some \( C \geq 1 \) such that

\[
\frac{1}{C} \|\cdot\| \leq \|\cdot\|_k \leq C \|\cdot\| \quad \text{for all} \ k \in \mathbb{N}.
\]

In practice \( \|\cdot\|_k \) is related to the Hessian of some smooth function at iteration point \( x_k \) and \( C \) is some (usually not explicitly known) bound of that Hessian.

Notice that the definition of \( f^0(x; h) \) and of \( \partial f(x) \) as subset of \( X^* \) merely uses convergence in \( X \) and, thus, does not depend on equivalent norms on \( X \). However the Riesz mapping identifying \( X^* \) with the Hilbert space \( X \) depends on the norm. Therefore \( \partial f(x) \) depends on the norm if it is considered as subset of \( X \), which we usually do for simplification of notation.
Remark 3.7. The gradient $\partial f(x)$ based on the norm $\| \cdot \|_k$ is understood as subset of $X$ equipped with $\| \cdot \|_k$ where, in particular, $B_\varepsilon(x)$ is taken with respect to $\| \cdot \|_k$.

3.1 Algorithm

Now we introduce the main algorithm based on two subalgorithms presented afterwards. We formulate several results that justify the single steps and that finally show convergence of the algorithm (cf. Figure 1 below for a rough overview). The proofs are collected in Section 3.2.

Algorithm 3.8 (Main Algorithm).

1. Initialization: Choose $T_1, T_2$ and $G$ satisfying Assumption 3.3,

   $\delta \in ]0, 1[,$ $x_0 \in X,$ $\varepsilon_0 > 0$

   and set $i = k = 0$.

2. Choose some norm $\| \cdot \|_k$ subject to Assumption 3.5, some 

   $a_k \in \partial f(x_k)$ (w.r.t. $\| \cdot \|_k$), and

   some $\varepsilon_{k,0} \geq G(\|a_k\|_k, \varepsilon_k)$.

3. Determine $a_{k,i} \in \partial \varepsilon_{k,i} f(x_k)$ (w.r.t. $\| \cdot \|_k$) by Algorithm 3.14 such that the null step condition

   $\|a_{k,i}\|_k < T_1(\varepsilon_{k,i})$ (3.9)

   or the condition of sufficient descent

   $f(x_k - \varepsilon_{k,i} h_{k,i}) - f(x_k) \leq -\delta \|a_{k,i}\|_k \varepsilon_{k,i}$ with $h_{k,i} := \frac{a_{k,i}}{\|a_{k,i}\|_k}$ (3.10)

   is satisfied (recall Remark 3.7 for the meaning of $\partial \varepsilon_{k,i} f(x_k)$ related to $\| \cdot \|_k$ and notice that (3.9) and (3.10) can be satisfied simultaneously).

4. In case (3.9) set $\varepsilon_{k,i+1} := T_2(\varepsilon_{k,i})$, increment $i$ by one, and go to Step 3.

5. If (3.9) is not true, choose $\sigma_k \geq \varepsilon_{k,i}$ such that the condition of sufficient descent

   $f(x_k - \sigma_k h_{k,i}) - f(x_k) \leq -\delta \|a_{k,i}\|_k \sigma_k$ (3.11)

   is satisfied (notice that $\sigma_k = \varepsilon_{k,i}$ is always possible, since (3.10) is satisfied in this case). Then fix the new iteration point

   $x_{k+1} := x_k - \sigma_k h_{k,i}$, (3.12)

   set $\varepsilon_{k+1} := \varepsilon_{k,i}$, increment $k$ by one, set $i = 0$, and go to Step 2.

Remark 3.13.

1. Instead of $\varepsilon_{k,0} \geq G(\|a_k\|_k, \varepsilon_k)$ in Step 2 one could also choose

   $\varepsilon_{k,0} \geq G(\|a_{k-1}\|_{k-1}, \varepsilon_k)$ for $k > 0$.

2. The selection of $\sigma_k$ in Step 5 can be done by some line search in direction $h_{k,i}$ (cf. Pytlak [26]).

3. One can easily ensure that $\sigma_k \to 0$ by requiring that e.g.

   $\varepsilon_k \leq \sigma_k = \|x_k - x_{k+1}\| \leq T_3(\varepsilon_k)$

   for some $T_3 : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ with $T_3(x) > x$ and $\lim_{t \to 0} T_3(t) = 0$, since the proof of Theorem 3.24 shows that $\varepsilon_k \to 0$. But in practice this is usually not necessary.
The essential point in Algorithm 3.8 is Step 3 with the computation of a suitable approximation $a_{k,i}$ of the norm-smallest element $\tilde{a}_{k,i} \in \partial^{k,i}f(x_k)$ (cf. (2.9)) such that the null step condition or the condition of sufficient descent is satisfied for given $\varepsilon_{k,i}$. Let us briefly discuss the main idea before we formulate the corresponding subalgorithm. Usually the sets $\partial f(y)$ defining $\partial^{k,i}f(x_k)$ are not known explicitly. For the algorithm we merely suppose that always at least one element $a \in \partial f(y)$ can be determined numerically (cf. Remark 3.19 below for a brief discussion of that point). On this basis we select step by step elements $b_j' \in \partial f(y_j)$ for suitable $y_j \in B_{\varepsilon_{k,i}}(x_k)$ and determine certain $a_j' \in \partial^{k,i}f(x_k)$ such that, roughly speaking, the convex hull $A_j'$ of all $a_j'$, $b_j'$ with $l \leq j$ is an approximating subset of $\partial^{k,i}f(x_k)$ and $a_j'$ is a norm-minimal element in $A_j'$. Hence, the null step condition (3.9) has to be satisfied for some $\varepsilon_{k,i}$.

In the last case $-h_{k,i} := -\frac{\partial f_k(x_k)}{\|\partial f_k(x_k)\|}$ is a descent direction on $B_{\varepsilon_{k,i}}(x_k)$ and, by Proposition 2.3 (4),

$$f(x_k - \varepsilon_{k,i}h_{k,i}) - f(x_k) \leq \varepsilon_{k,i}f_0(x_k - h) \leq -\delta \varepsilon_{k,i}\|a_{k,i}\|$$

with $\delta \in ]0,1[$ from Algorithm 3.8 i.e. condition (3.10) of sufficient descent is satisfied with the standard norm. Clearly the quality of the algorithm is closely related to the quality of the approximating set $A_j'$ and, in some applications, we can improve the quality substantially by choosing a suitable equivalent norm $\|\cdot\|_k$ in each step.

Let us now provide the precise algorithm where quantities determined here are marked by $'$. **Algorithm 3.14.** Let $T_1$, $\delta \in ]0,1[$, $x_k \in X$, $\varepsilon_{k,i} > 0$, and $\|\cdot\|_k$ be as in Step 3 of Algorithm 3.8 for some $k, i \in \mathbb{N}$.

1. Choose some $a_0' \in \partial f(x_k)$ (w.r.t. $\|\cdot\|_k$) and some $\delta' \in ]0,1[$ and set $j = 0$. (Typically, but not necessarily, $a_0'$ agrees with $a_k$ from Algorithm 3.8)

2. Set $a_{k,i} := a_j'$ and $h_j' := \frac{a_j'}{\|a_j'\|}$. If $a_{k,i}$ satisfies the null step condition (3.9) or condition (3.10) of sufficient descent, stop and return $a_{k,i}$.

3. Otherwise compute some $b_j' \in \partial f(y_j)$ (w.r.t. $\|\cdot\|_k$) for some $y_j \in [x_k, x_k - \varepsilon_{k,i}h_j']$ by Algorithm 3.17 such that

$$\langle a_j', b_j' \rangle_k \leq \delta' \|a_j'\|^2_k.$$  

4. Choose some subset $B_j' \subseteq \{a_j' \mid l \leq j\} \cup \{b_j' \mid l \leq j\}$ such that $a_j', b_j' \in B_j'$ and set

$$A_j' := \text{conv } B_j'.$$

5. Compute

$$a_{j+1}' := \arg \min \{\|a'\|_k \mid a' \in A_j'\},$$

increment $j$ by one, and go to Step 2.

Notice that $a_j', b_j' \in \partial^{k,i}f(x_k)$ for all $1 \leq l \leq j$ by induction and that

$$A_j' = \text{conv } B_j' \subseteq \partial^{k,i}f(x_k).$$

Hence $A_j'$ can be considered as some inner approximation of $\partial^{k,i}f(x_k)$ and the norm-smallest element $a_{j+1}' \in A_j'$ is an approximation of the norm-smallest element $\tilde{a}_{k,i} \in \partial^{k,i}f(x_k)$. Algorithm 3.14 ensures with (3.15) that $\|a_j'\|$ decreases sufficiently, i.e. we have $\|a_j'_{l+1}\|_k \leq \gamma \|a_j'\|_k$ for some $\gamma \in (0,1)$ as long as the null step condition (3.9) is not fulfilled (cf. the proof of Lemma 3.30). Hence, the null step condition (3.9) has to be satisfied for some $a_{k,i} = a_j'$ after
finitely many steps if we do not meet condition \( (3.10) \) of sufficient descent before. In practice we usually take

\[
B'_j = \{a'_0\} \cup \{a'_l \mid j - m \leq l \leq j\} \cup \{b'_j\} \quad \text{or} \quad B'_j = \{a'_0, a'_j\} \cup \{b'_l \mid j - m \leq l \leq j\}
\]

with \( m \approx 10 \).

**Remark 3.16.** Note that the computation of \( a'_{j+1} \) is equivalent to the minimization of a quadratic function defined on some \#\( B'_j \)-simplex. This can be easily done with SQP or semi smooth Newton methods (cf. [11, 24, 29]). Since \( \dim X \gg m \) for typical applications, we can neglect the computational time for \( a'_{j+1} \) compared to that needed for the computation of a gradient of \( f \).

We complete our algorithm with the precise scheme about the selection of \( y_{ij} \) in Step 3 of Algorithm 3.14 by some nesting procedure for the segment \([x_k, x_k - \varepsilon_{k,i} h'_{j}]\). New quantities determined in the subalgorithm are marked by "\( ^\prime\)".

**Algorithm 3.17.** Let \( 0 < \delta < \delta' < 1, x_k \in X, \varepsilon_{k,i} > 0, \|\cdot\|_k, a'_k \in \partial^{k,i} f(x_k), \) and \( h'_{j} \) be as in Step 3 of Algorithm 3.14 (Notice that both the null step condition \( (3.9) \) and condition \( (3.10) \) of sufficient descent are violated for \( a_{k,i} = a'_{j,j} \)).

1. Set \( l = 0, x'_0 := x_k, \) and \( y'_0 := x_k - 2\varepsilon_{k,i} h'_{j} \) (notice that \( x'_0 + y'_0 \over 2 = x_k - \varepsilon_{k,i} h'_{j} \)).
2. Choose some \( b'_j \in \partial(x'_0 + y'_0) \) (w.r.t. \( \|\cdot\|_k \)).
3. If \( b'_j := b'_0 \) satisfies \( (3.15) \) stop and return \( b'_j \).
4. Otherwise choose \( x''_{l+1} \in \{x''_l, {x''_l + y''_l \over 2}\} \) and \( y''_{l+1} \in \{x''_{l+1}, y''_{l+1}\} \) such that

\[
\|y''_{l+1} - x''_{l+1}\|_k = {1 \over 2} \|y''_{l} - x''_{l}\|_k \quad \text{and} \quad f(y''_{l+1}) - f(x''_{l+1}) > -\delta \|a'_j\|_k \|y''_{l+1} - x''_{l+1}\|_k
\]

(3.18)

where we take \( x''_0 = x'_0 \) if \( l = 0 \) (this way the condition of sufficient descent is violated on segment \([x''_{l+1}, y''_{l+1}]\) with \( a = a'_{j,j} \)).
5. Increment \( l \) by 1 and go to Step 2.

A slightly simplified survey about the complete algorithm is given in Figure 1

**Remark 3.19.** While the implementation of the most steps in Algorithm 3.18 and its subalgorithms should be quite clear, let us briefly discuss how to choose some element \( a \in \partial f(x) \). In our applications we usually have a representation of \( \partial f(x) \) that allows the numerical computation of some element \( a \in \partial f(x) \). More precisely, in many cases \( f \) is continuously differentiable on an open set \( U \subset X = \mathbb{R}^n \) such that \( \mathbb{R}^n \setminus U \) has zero Lebesgue measure. Here we can use Proposition 2.1.5 or Theorem 2.5.1 from Clarke [5] to get single elements \( a \in \partial f(x) \). If \( f \) is defined to be the pointwise maximum or minimum of smooth functions, Proposition 2.3.12 or Proposition 2.7.3 in [5] can be used to determine some \( a \in \partial f(x) \). Moreover we can combine this with other calculus rules as e.g. the chain rule [8, Theorem 2.3.9]. Beyond these methods, that are sufficient for the benchmark problems considered in Section 2, also discrete approximations of elements of \( \partial f(x) \) like e.g. in [3] can be used. Let us finally state that the presented algorithm assumes the possibility to compute at least one element of \( \partial f(x) \).

Let us now justify the essential steps of the algorithm, i.e. that the required conditions can be reached and that the iterations typically terminate after finitely many steps. We start with Algorithm 3.17 and consider in particular Step 4.
choose a norm \( \| \cdot \|_k \), some \( a_k \in \partial f(x_k) \) (w.r.t. \( \| \cdot \|_k \)), some \( \varepsilon_{k,0} \geq G(\|a_k\|_k, \varepsilon_k) \), and \( i := 0 \)

\[ k := k + 1 \]

\[ i := i + 1 \]

\[ j := 0, \text{ choose } a'_0 \in \partial f(x_k) \text{ (w.r.t. } \| \cdot \|_k, \text{ e.g. } a'_0 := a_k) \]

\[ j := j + 1 \]

null step: \( l \in \sigma^{k-1} f(x_k) \)

\[ \|a'_j\|_k \leq T_1(\varepsilon_{k,i}) \]

\[ \text{NO} \]

sufficient descent

\[ h'_j := \frac{a'_j}{\varepsilon'_j} \]

\[ f(x_k - \varepsilon_k h'_j) - f(x_k) \leq -\delta||a'_j||_k \varepsilon_{k,i} \]

\[ \text{YES} \]

\[ \text{YES} \]

\[ \text{NO} \]

Algorithm 3.17

(preconditions step size)

\[ \text{Algorithm 3.17} \]

compute \( b'_j \in \sigma^{k-1} f(x_k) \) with

\[ \langle a'_j, b'_j \rangle \leq \delta||a'_j||_k^2 \]

with inner approximation of \( \partial f(x_k) \)

\[ A'_j := \text{conv} \{ \{a'_l | l < j\} \cup \{h'_l | l < j\} \} \]

improve norm-minimal element \( a'_j \) by

\[ a'_{j+1} := \arg \min \{ ||a'||_k | a' \in A'_j \} \]

null step: \( \varepsilon_{k,i+1} := T_2(\|a_{k,i}\|_k) < \varepsilon_{k,i} \)

descent step: line search in direction \(-h_{k,i}\) for step size \( \sigma_k > \varepsilon_k \) and

\[ x_{k+1} := x_k - \sigma_k h_{k,i} \]

Figure 1: Flow diagram of Algorithm 3.8

**Proposition 3.20** (properties of Algorithm 3.17). Let the assumptions of Algorithm 3.17 be satisfied. Then:

1. The choice in Step 4 of Algorithm 3.17 is possible for every \( l \in \mathbb{N} \).
2. The set

\[ \Lambda_l := \{ \lambda \in [0, 1] \mid \text{there is some } b'_j \in \partial f(\lambda x''_l + (1 - \lambda)y''_l) \text{ satisfying (3.15)} \} \]

has positive Lebesgue measure for every \( l \in \mathbb{N} \).
3. If Algorithm 3.17 does not terminate and, therefore, produces sequences \( (x''_l) \) and \( (y''_l) \) converging to some \( y''_\infty \in [x''_l, y''_l] \), then there is some \( b'_l \in \partial f(y''_\infty) \) satisfying (3.15) and \( f \) is not strictly differentiable at \( y''_\infty \).
4. If \( f \) is convex on a neighborhood of \( [x''_l, y''_l] \), then Algorithm 3.17 terminates in Step 3 already for \( l = 0 \).

Though it is not trusted that we find some \( b'_l \) satisfying (3.15) after finitely many steps, there is an extremely good chance according to Proposition 3.20(2). In practice Algorithm 3.17 always terminated, also for rather complex simulations presented here. Nevertheless there are examples where the algorithm does not terminate (at least theoretically) as a simple induction argument shows, e.g., for \( f(t) = -t^2 \sin(2\pi t) \) with \( \varepsilon_{k,i} = 1 = y''_l = a'_j \) and \( f(0) = 0 = x''_l \).
Remark 3.21. Typically it is much cheaper for time consumption to compute merely the scalar $(b'_j, a'_j)_k$ in (3.13) instead of the complete vector $b'_j$. Therefore we compute $b'_j$ only if (3.15) is satisfied.

Proposition 3.22 (properties of Algorithm 3.14). Let the assumptions of Algorithm 3.14 be satisfied, let $f$ be Lipschitz continuous on some neighborhood of $B_{\varepsilon k,i}(x_k)$, and suppose that Algorithm 3.17 always terminates. Then Algorithm 3.14 stops after finitely many steps and returns some $a_{k,i} \in \partial^{\varepsilon_k} f(x_k)$ satisfying (3.9) or (3.10).

Proposition 3.23 (properties of Algorithm 3.8). Let the assumptions of Algorithm 3.8 be satisfied and let $x_k$ be an iteration point from that algorithm. Then:

1. If $\varepsilon_{k,i} > 0$ is related to $x_k$ for some $i \in \mathbb{N}$, then there exists $a_{k,i} \in \partial^{\varepsilon_k} f(x_k)$ satisfying the null step condition (3.9) or condition (3.10) of sufficient descent.
2. If $0 \notin \partial f(x_k)$, then there are only finitely many $i \in \mathbb{N}$ such that (3.9) is satisfied.

Though Proposition 3.23 (1) already follows from Propositions 3.22 we will still give a brief independent proof of it in the next section.

Summarizing we can say that, in principle, the presented algorithm always works and cannot get stuck, i.e. at most finitely many subiterations are necessary to find a new iteration point $x_{k+1}$. The only point is that Algorithm 3.17 might not terminate which, however, is quite unlikely according to Proposition 3.20 (2) and which never happened in our simulations.

Let us finally confirm that the presented descent algorithm can reach both minimizers and critical points of $f$.

Theorem 3.24 (accumulation points are critical points). Let the assumptions of Algorithm 3.8 be satisfied and let $x_k$ be a corresponding sequence of iteration points. Then $(f(x_k))$ is strictly decreasing. Moreover, if $x$ is an accumulation point of $(x_k)$, then $0 \in \partial f(x)$ and $f(x_k) \to f(x)$.

As consequence we can formulate some more precise statement.

Proposition 3.25. Let the assumptions of Algorithm 3.8 be satisfied, let $(x_k)$ be a corresponding sequence of iteration points with step sizes $\sigma_k \to 0$, and suppose that $\{x_k \mid k \in \mathbb{N}\}$ is relatively compact. Then each accumulation point of $(x_k)$ is a critical point of $f$ and, if \{\(x \in X \mid f(x) \leq f(x_0)\)\} contains only finitely many critical points, $(x_k)$ converges to a critical point of $f$. Moreover, if $(x_k)$ is not convergent, then $(x_k)$ has no isolated accumulation point.

Remark 3.26. If $X = \mathbb{R}^n$ and $\{x \in X \mid f(x) \leq f(x_0)\}$ is bounded, then $\{x_k \mid k \in \mathbb{N}\}$ is relatively compact.

3.2 Proofs

Proof of Proposition 3.20. Since $a_{k,i} := a'_j$ does not satisfy (3.9), we have $\|a'_j\|_k \neq 0$. By construction $\|y_l - x''_l\|_k = (\frac{1}{2})^{l-1} \varepsilon_{k,i} > 0$.

(1) Since (3.10) is not fulfilled, we have (3.18) for $l = 0$ with $y''_l = x_k - \varepsilon_{k,i} b'_j$. Assume that (3.18) holds for $l - 1$. Then

$$f(x'_l) - f\left(\frac{x''_l + y''_l}{2}\right) + f\left(\frac{x''_l + y''_l}{2}\right) - f(x''_l) > -\delta \|a'_j\|_k \|x''_l - y''_l\|_k. \quad (3.27)$$

If neither $x''_{l+1} = x'_l$ and $y''_{l+1} = \frac{x''_l + y''_l}{2}$ nor $x''_{l+1} = \frac{x''_l + y''_l}{2}$ and $y''_{l+1} = y''_l$ satisfy (3.18) for $l$, then

$$f(x'_l) - f\left(\frac{x''_l + y''_l}{2}\right) + f\left(\frac{x''_l + y''_l}{2}\right) - f(y''_l) \leq -2\delta \|a'_j\|_k \frac{1}{2} \|x''_l - y''_l\|_k$$
which contradicts (3.27). Hence the choice of \( x''_{l+1} \) and \( y''_{l+1} \) is always possible and induction gives (3.18) for all \( l > 0 \).

(2) Consider \( f_t : [0,1] \to \mathbb{R} \) with \( f_t(t) := f(y''_t + t(x''_t - y''_t)) \). As Lipschitz continuous function, \( f_t \) is absolutely continuous and differentiable almost everywhere. Therefore

\[
\frac{1}{t} \int_0^t f'(t) dt = f(x''_t) - f(y''_t) \geq -\delta \|a'_j\|_k \|x''_t - y''_t\|_k,
\]

Hence the set of all \( t \in [0,1] \) with \( f'_t(t) > -\delta \|a'_j\|_k \|x''_t - y''_t\|_k \) has positive measure. Using Clarke’s chain rule \( [8, \text{Theorem 2.3.10}] \) and \( x''_t - y''_t = -\|x''_t - y''_t\|_k \|a'_j\|_k \), we get

\[
\dot{f}'(t) \in \langle \dot{\partial}f(y''_t + t(x''_t - y''_t)), x''_t - y''_t \rangle_k \frac{-\|x''_t - y''_t\|_k}{\|a'_j\|_k} \langle \dot{\partial}f(y''_t + t(x''_t - y''_t)), a'_j \rangle_k
\]

which implies that \( \Lambda_t \) has positive Lebesgue measure.

(3) By Lebourg’s mean value theorem (cf. \( [8, \text{Prop. 2.3.7}] \)), there exists some \( \tilde{x''}_t \in [x''_t, y''_t] \) and some \( b''_t \in \dot{\partial}f(\tilde{x''}_t) \) with

\[
-\delta \|a'_j\|_k \|y''_t - x''_t\|_k \leq f(x''_t) - f(y''_t) = (b''_t, x''_t - y''_t)_k = -\|y''_t - x''_t\|_k \|a'_j\|_k (b''_t, a'_j)_k.
\]

Clearly \( \frac{\tilde{x''}_t + y''_t}{2} \to y''_\infty \) and \( \tilde{x''}_t \to y''_\infty \). Since \( f \) is Lipschitz continuous near \( y''_\infty \), the sequences \( (b''_t) \) and \( (b''_t) \) are bounded in the Hilbert space \( X \). Therefore, up to a subsequence, \( b''_t \to b'' \) and \( \tilde{b''}_t \to : b''_t \). The upper semicontinuity of Clarke’s generalized gradient implies \( b''_t, b''_t \in \dot{\partial}f(y''_\infty) \) (cf. \( [8, \text{Prop. 2.1.5}] \)). Since we assume that (3.15) always fails in Step 3 of Algorithm 3.17 we have

\[
\dot{\partial}f(y''_\infty)(b''_t, a'_j)_k < \delta \|a'_j\|_k^2 \delta \|a'_j\|_k^2 < (b''_t, a'_j)_k.
\]

Taking the limit we obtain that \( b''_t \) satisfies (3.15) and that \( b''_t \neq b'' \). Hence \( \dot{\partial}f(y''_\infty) \) is not a singleton and, consequently, \( f \) is not strictly differentiable at \( y''_\infty \) (cf. \( [8, \text{Prop. 2.2.2}] \)).

(4) Recall that Clarke’s generalized gradient \( \dot{\partial}f(x) \) agrees with the (convex) subdifferential for a convex \( f \) (cf. \( [8, \text{Prop. 2.2.7}] \)). Since (3.10) is not satisfied for \( a_{k,i} = a_j \in \dot{\partial}f(x''_0) \), the definition of the subdifferential gives for any \( b''_0 \in \dot{\partial}f \left( \frac{x''_0 + y''_0}{2} \right) \) that

\[
\delta \|a'_j\|_k \|b''_0\|_k \geq f(x''_0) - f \left( \frac{x''_0 + y''_0}{2} \right) \geq \langle b''_0, a'_j \rangle_k.
\]

which implies (3.15) for \( b''_t = b''_0 \).\( \diamond \)

As preparation for the proof of Proposition 3.22 we consider some technical lemma. Notice that the statement remains true with \( \| \cdot \|_k \) instead of \( \| \cdot \| \).

**Lemma 3.30.** Let \( \gamma \in [0,1] \) and \( L > 0 \) be constants and let \( (a'_j) \) and \( (b'_j) \) be sequences in \( X \) such that for all \( j \in \mathbb{N} \):

1. \( \langle a'_j, b'_j \rangle \leq \gamma \|a'_j\|^2 \),
2. \( \|b'_j\| \leq L \),
3. \( \|a'_{j+1}\| \leq \min_{\lambda \in [0,1]} \|\lambda a'_j + (1 - \lambda)b'_j\| \).

Then \( \|a'_j\| \to 0 \).
Proof. The sequence \((\|a_j'\|)\) is non increasing by (3) and, hence, convergent. Since \(L\) is just a bound for the \(b_j'\) according to (2), we can assume that \(\|a_j'\| < L\) for all \(j \in \mathbb{N}\). Therefore \((1 - \gamma)\|a_j'\|^2 < L^2\) and

\[
\lambda_j := 1 - \frac{(1 - \gamma)\|a_j'\|^2}{(1 - \gamma)^2\|a_j'\|^2 + L^2} \in [0, 1].
\] (3.31)

We derive

\[
\|a_{j+1}'\|^2 \leq \|\lambda_j a_j' + (1 - \lambda_j)b_j'\|^2 \\
= \lambda_j^2\|a_j'\|^2 + 2\lambda_j(1 - \lambda_j)\langle a_j', b_j' \rangle + (1 - \lambda_j)^2\|b_j'\|^2 \\
\leq (\lambda_j^2 + 2\lambda_j(1 - \lambda_j)\gamma\|a_j'\|^2 + (1 - \lambda_j)^2L^2 \\
\leq (\lambda_j + (1 - \lambda_j)\|a_j'\|^2 + (1 - \lambda_j)^2L^2 \\
\|a_j'\|^2 \\
= \frac{L^2\|a_j'\|^2}{(1 - \gamma)^2\|a_j'\|^2 + L^2} \\
\frac{L^2(1 - \gamma)^2\|a_j'\|^4}{(1 - \gamma)^2\|a_j'\|^2 + L^2} \\
\frac{L^2\|a_j'\|^2}{(1 - \gamma)^2\|a_j'\|^2 + L^2}.
\]

Taking the limit we obtain with \(\alpha := \lim_{j \to \infty} \|a_j'\|^2\) that \(\alpha \leq \frac{\alpha L^2}{(1 - \gamma)^2\alpha + L^2}\). But this is only possible for \(\alpha = 0\), which implies the assertion. \(\diamondsuit\)

Proof of Proposition 3.22. Let \(L\) be the Lipschitz constant of \(f\) on some neighborhood of \(B_{\epsilon_{k,i}}(x_k)\). Thus the \(b_j'\) are bounded by \(L\) according to Proposition 2.23. With (3.15) and the definition of \(a_j'\) we verify the assumptions of Lemma 3.30 with \(\gamma = \delta'\) and the norm \(\|\cdot\|\). We obtain the claim, since the algorithm stops as soon as \(\|a_j'\| \leq T_1(\epsilon_{k,i})\). \(\diamondsuit\)

Proof of Proposition 3.23. For the first assertion we fix \(k, i \in \mathbb{N}\). If \(0 \in \partial^{\epsilon_{k,i}}f(x_k)\) (w.r.t. \(\|\cdot\|\)), then \(a_{k,i} := 0\) satisfies (3.9). Otherwise we choose \(a_{k,i} := a, h_{k,i} = -h\) with \(a, h\) as in Proposition 2.28 for \(x = x_k\) and \(\epsilon = \epsilon_{k,i}\). By Lebourg’s mean value theorem (cf. [8, Prop. 2.3.7]) there are \(y \in [x_k, x_k - \epsilon_{k,i}h_{k,i}]\) and \(a, h \in \partial f(y) \subseteq \partial^{\epsilon_{k,i}}f(x_k)\) (w.r.t. \(\|\cdot\|\)) such that

\[
f(x_k - \epsilon_{k,i}h_{k,i}) - f(x_k) = \epsilon_{k,i}(a, -h_{k,i})k = \epsilon_{k,i}(a, h)k \leq \epsilon_{k,i}f_{\epsilon_{k,i}}^0(x; h) < -\delta\epsilon_{k,i}\|a_{k,i}\|k.
\] (2.10)

Thus (3.10) is satisfied and the claim is verified. (Notice that the statement also follows from Proposition 3.20 and Proposition 3.22 if the corresponding assumptions are satisfied.)

For the second assertion we fix \(k\) and \(\|\cdot\|\). Since \(0 \notin \partial f(x_k)\), Lemma 2.6 provides \(\epsilon, K > 0\) such that

\[
\|a\|_k > K \quad \text{for all } a \in \partial f(x_k).
\]

If the algorithm would make infinitely many null steps, i.e. (3.9) holds for infinitely many \(i \in \mathbb{N}\), then \(\epsilon_{k,i} \to 0\) and \(T_1(\epsilon_{k,i}) \to 0\) by \(\epsilon_{k,i+1} = T_2(\epsilon_{k,i})\) and Assumption 3.3. Hence \(\epsilon_{k,i} < \epsilon\) for large \(i\) and, thus,

\[
0 < K \leq \|a_{k,i}\|_k \leq T_1(\epsilon_{k,i}) \to 0.
\] (3.9)
But this is a contradiction and verifies the assertion.

\[ (*) \]

**Proof of Theorem 3.24** \((f(x_k))\) is strictly decreasing by construction (cf. (3.11)). Let us suppose that \(x\) is an accumulation point of \((\|x_k\|)\). Then \((f(x_k))\) has a unique accumulation point and, by continuity of \(f\), we have \(f(x_k) \to f(x)\).

By \(i_k\) we denote the index \(i\) related to \(k\) leading to the new iteration point \(x_{k+1}\) in Step 5 of Algorithm 3.8 (notice that the assumptions of the theorem imply the existence of \(i_k\)). Then we have \(\varepsilon_{k+1} = \varepsilon_{k,i_k}\) and we set \(\hat{a}_k := a_{k,i_k}\). Since the null step condition (3.9) is never satisfied for \(a_{k,i_k}\) and since \(\sigma_k \geq \varepsilon_{k,i_k}\), we have for all \(N \in \mathbb{N}\)

\[
f(x) - f(x_0) \leq f(x_{N+1}) - f(x_0) = \sum_{k=0}^{N} f(x_{k+1}) - f(x_k) \tag{3.11}
\]

\[
\leq \sum_{k=0}^{N} -\delta \sigma_k \|\hat{a}_k\| \leq -\delta \sum_{k=0}^{N} \varepsilon_{k+1} T_1(\varepsilon_{k+1}) .
\]

Hence the right hand side is bounded independent of \(N\). Therefore \(T_1(\varepsilon_k) \varepsilon_k \to 0\) and, since \(T_1\) is nondecreasing,

\[
\varepsilon_k \to 0 . \tag{3.32}
\]

For contradiction we assume that \(0 \notin \partial f(x)\). By Lemma 2.6 we find \(\varepsilon > 0\) and \(h \in X\) with \(\|h\| = 1\) such that

\[
f_0^h(x; h) < 0 \quad \text{(w.r.t. } \|\cdot\|) .
\]

With \(C \geq 1\) from Assumption 3.5 we have for all \(a \in \partial^{k,i} f(x_k)\) (w.r.t. \(\|\cdot\|\))

\[
\|a\| \geq 1 - \frac{\|h\|}{C} \geq \frac{1}{C} (a, h) \tag{2.24} \geq \frac{1}{C} f_0^h(x; h) = \frac{1}{C} \sup_{y \in X: \|y - x_k\| \leq \varepsilon_{k,i}} f^0(y; h) .
\]

For \(k, i \in \mathbb{N}\) with

\[
\|x_k - x\| < \frac{\varepsilon}{2} \quad \text{and} \quad \varepsilon_{k,i} < \frac{\varepsilon}{2C} \tag{3.34}
\]

and all \(y \in X\) with \(\|y - x_k\| \leq \varepsilon_{k,i}\) we have

\[
\|y - x\| \leq \|y - x_k\| + \|x_k - x\| \leq C \|y - x_k\| + \frac{\varepsilon}{2} \leq C \varepsilon_{k,i} + \frac{\varepsilon}{2} \leq \varepsilon .
\]

Therefore

\[
\sup_{y \in X: \|y - x_k\| \leq \varepsilon_{k,i}} f^0(y; h) \leq \sup_{y \in X: \|y - x\| \leq \varepsilon} f^0(y; h) \tag{2.22} \leq f_0^0(x; h) .
\]

Consequently (3.33) gives

\[
\|a\| \geq -\frac{1}{C} f_0^0(x; h) =: K > 0 \tag{3.35}
\]

for all \(a \in \partial^{k,i} f(x_k)\) (w.r.t. \(\|\cdot\|\)) and all \(k, i\) satisfying (3.34).

(i) As a first case we assume that there is some \(k_0 \in \mathbb{N}\) with \(\|x_k - x\| < \frac{\varepsilon}{2}\) for all \(k \geq k_0\). Recall that

\[
\varepsilon_{k,0} \geq G(\|a_k\|, \varepsilon_k) .
\]

If \(k \geq k_0\) and \(i_k \geq 0\), then (3.3) is valid for \(i = i_k - 1\) and, with (3.35),

\[
T_1(\varepsilon_{k,i_k-1}) > K \quad \text{if} \quad \varepsilon_{k,i_k-1} < \frac{\varepsilon}{2C} . \tag{3.36}
\]
By Assumption 3.3 there is \( \varepsilon_K > 0 \) such that \( T_1(t) \leq K \) for all \( t \leq \varepsilon_K \). Therefore
\[
\varepsilon_{k,i_k-1} \geq \min \{ \varepsilon_K, \frac{\varepsilon}{2C} \} =: \bar{\varepsilon} > 0 \quad \text{for} \quad k \geq k_0
\]
(otherwise \( K \geq T_1(\varepsilon_K) \geq T_1(\varepsilon_{k,i_k-1}) > K \) by (3.30)). Since \( T_2 \) is not decreasing,
\[
\varepsilon_{k,i_k} = T_2(\varepsilon_{k,i_k-1}) \geq T_2(\bar{\varepsilon}) \quad \text{if} \quad i_k > 0 \text{ and } k \geq k_0.
\]
Thus, for general \( i_k \) and \( a_k \in \partial f(x_k) \) from Step 2 of Algorithm 3.8,
\[
\varepsilon_{k+1} = \varepsilon_{k,i_k} \geq \min \{ \varepsilon_k, T_2(\bar{\varepsilon}) \} \geq \min \{ G(\|a_k\|_k, \varepsilon_k), T_2(\bar{\varepsilon}) \} \quad \text{for} \quad k \geq k_0.
\]
From \( \varepsilon_k \to 0 \) we obtain \( G(\|a_k\|_k, \varepsilon_k) \to 0 \). Then we can discuss the two cases of Assumption 3.3 for \( G \) separately:

(a) We get \( \|a_k\|_k \to 0 \). This contradicts (3.35), since \( \varepsilon_{k+1} = \varepsilon_{k,i_k} < \frac{\varepsilon}{2C} \) and \( a_k \in \partial f^{k,i_k} f(x_k) \) for all \( k \) large enough.

(b) Using (3.35) we have \( \varepsilon_{k+1} \geq G(\|a_k\|_k, \varepsilon_k) \geq \varepsilon_k \) for all \( k \) sufficiently large. But this contradicts \( \varepsilon_k \to 0 \).

The contradictions imply that \( 0 \in \partial f(x) \) in the special case (i). Notice that we can argue analogously in the case of Remark 3.13.

(ii) Now we assume that \( \|x_k - x\| \geq \frac{\varepsilon}{2} \) for infinitely many \( k \). Since \( x \) is an accumulation point of \( \{x_k\} \), we can choose a subsequence \( \{x_{k(i)}\} \), such that
\[
\varepsilon_k < \frac{\varepsilon}{2C} \quad \text{for all} \quad k > k(0),
\]
\[
\|x_{k(2j)} - x\| < \frac{\varepsilon}{4}, \quad \|x_{k(2j+1)} - x\| \geq \frac{\varepsilon}{2}
\]
and
\[
\|x_l - x\| < \frac{\varepsilon}{2} \quad \text{for} \quad k(2j) \leq l < k(2j + 1).
\]
Using \( I_N := \{ j \in \mathbb{N} \mid k(2j + 1) < N \} \) and \( f(x_{k+1}) \leq f(x_k) \) we get
\[
f(x) - f(x_0) = \lim_{N \to \infty} f(x_N) - f(x_0)
\leq \lim_{N \to \infty} \sum_{j \in I_N} f(x_{k(2j)}) - f(x_{k(2j)})
= \lim_{N \to \infty} \sum_{j \in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} f(x_{l+1}) - f(x_l)
\leq -\delta K \lim_{N \to \infty} \sum_{j \in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} \|x_{l+1} - x_l\|_l
\leq -\delta K \lim_{N \to \infty} \sum_{j \in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} \|x_{l+1} - x_l\|_l
\leq -\delta K \lim_{N \to \infty} \sum_{j \in I_N} \|x_{k(2j+1)} - x_{k(2j)}\|_l
\]
14
\[-\frac{\delta K}{C} \lim_{N \to \infty} \sum_{j \in I_N} \varepsilon = -\infty\]

which is impossible.

Hence we have a contradiction in both cases (i) and (ii). Therefore \(0 \in \partial f(x)\) and the assertion is verified. \(\triangle\)

**Proof of Prop. 3.25.**

By Theorem 3.24 accumulation points are critical points. Finite sets consist only of isolated points. The rest follows from general properties of accumulation points stated in the subsequent Proposition 3.37. \(\triangle\)

**Proposition 3.37.**

Let \((x_k)\) be a sequence in the Hilbert space \(X\) such that \(\{x_k \mid k \in \mathbb{N}\}\) is relatively compact and let \(A\) be the set of all accumulation points of \((x_k)\). Then \(A \neq \emptyset\) and

\[\text{dist}(x_k, A) := \inf \left\{ \|y - x_k\| \mid y \in A \right\} \to 0 \quad \text{as} \quad k \to \infty.\]

If in addition \(\|x_k - x_{k+1}\| \to 0\) as \(k \to \infty\), then either

\[A = \{x\} \quad \text{for some} \quad x \in X \quad (i.e. \quad x_k \xrightarrow{k \to \infty} x)\]

or \(A\) has no isolated points.

**Proof.** Clearly \(A \neq \emptyset\) by relative compactness. Let us now assume that \(\text{dist}(x_k, A) \to 0\). Then there is some \(K > 0\) and a subsequence \((x_{k(i)})_i\) with \(\text{dist}(x_{k(i)}, A) > K\) for all \(i \in \mathbb{N}\) which contradicts relative compactness.

Under the additional condition \(\|x_k - x_{k+1}\| \to 0\) we assume that \(A\) has an isolated point \(x\), i.e. there is some \(r > 0\) with \(B^r(x) \cap A = \{x\}\). Since \(x\) is accumulation point of \((x_k)\), there exists some \(k_0 \in \mathbb{N}\) with

\[x_{k_0} \in B^r(x) \quad \text{and} \quad \|x_k - x_{k+1}\| < \frac{r}{4} \quad \text{for all} \quad k \geq k_0.\]

Moreover, for a possibly larger \(k_0\), we have

\[\text{dist}(x_k, A) < \frac{r}{4} \quad \text{for all} \quad k \geq k_0,\]

since otherwise there is a subsequence \((x_{k(i)})\) with \(\text{dist}(x_{k(i)}, A) \geq \frac{r}{4}\) for all \(i\), which is impossible by relative compactness. We thus conclude that \(x_{k_0+1} \in B^r(x)\) and, by induction, \(x_k \in B^r(x)\) for all \(k \geq k_0\). Consequently \(x\) is the only accumulation point of \((x_k)\). \(\triangle\)

## 4 Benchmark problems

We now consider several classical benchmark problems on \(\mathbb{R}^n\) and compare our numerical results based on Algorithm 3.8 with that of other algorithms found in the literature. In our algorithm we normally choose

\[\delta' = 0.35, \quad \delta = 0.3 \quad \text{and} \quad T_2(x) = 0.35 \cdot x.\]  

(4.1)

Recall that the norm can be changed in each iteration step. We mainly consider the following two specializations of Algorithm 3.8 (briefly called Algorithm 3.8.A and 3.8.B):

\[
\leq -\delta K \lim_{N \to \infty} \sum_{j \in I_N} \varepsilon = -\infty
\]
(A) We do not change the norm, i.e. \( \| \cdot \|_k := \| \cdot \| \) in every iteration step \( k \) (typically \( \| \cdot \| \) is the Euclidean norm). Furthermore we set (if nothing else is stated)
\[
G(x, y) = y, \quad T_1(x) = \frac{x}{\varepsilon_0}
\]
where \( \varepsilon_0 \) is the initial radius of the initial neighborhood \( B_{\varepsilon_0}(x_0) \).

(B) We adapt the norm \( \| \cdot \|_k := \| A_k \cdot \| \) in each iteration \( k \) with some suitable symmetric and positively definite matrix \( A_k \). Moreover we choose
\[
G(x, y) = x, \quad T_1(x) < x.
\]

In Step 2 of Algorithm 3.8 we first determine some \( \tilde{a}_k \in \partial f(x_k) \) with respect to the standard norm \( \| \cdot \| \). Then we set
\[
a_k := A_k^{-2} \tilde{a}_k \quad \text{and} \quad \varepsilon_{k,0} := \| a_k \|_k
\]
where \( a_k \in \partial f(x_k) \) w.r.t. \( \| \cdot \|_k \), since for every \( h \in X \)
\[
\langle a_k, h \rangle_k = \langle A_k a_k, A_k h \rangle = \langle A_k A_k^{-2} \tilde{a}_k, A_k h \rangle = \langle \tilde{a}_k, h \rangle.
\]

In Step 5 we take \( \sigma_k \geq \varepsilon_k \geq \varepsilon_{k,i} \) and Algorithm 3.14 always starts with \( a'_0 = a_k \).

The benchmark functions \( f : \mathbb{R}^n \to \mathbb{R} \) considered for minimization are of course locally Lipschitz continuous. Some of them are even smooth, but show similar numerical difficulties as nonsmooth problems due to large second derivatives. In all cases we have an explicit formula for the computation of at least one element of the generalized gradient (cf. Remark 3.19).

We will compare our numerical results with results of the bundle method (BM) and the bundle trust region method (BTR) (cf. Alt [1, 2]), since these methods for convex functions inspired to some extend the development of our algorithm for (not necessarily convex) locally Lipschitz continuous functions. Furthermore we consider the BFGS algorithm\(^2\) which has been recently studied e.g. by Lewis and Overton. They minimized also nonsmooth functions and got promising results (cf. [19, 20]). In addition we compare with the gradient sampling algorithm (GS). Since a benchmark function \( f \) is typically treated by different algorithms in the literature, we compare all algorithms for \( f \) fixed.

### 4.1 Wolfe function

The Wolfe function \( f : \mathbb{R}^2 \to \mathbb{R} \) is a classical convex benchmark function given by
\[
f(x, y) := \begin{cases} 
9x + 16|y| - x^9 & \text{if } x \leq 0, \\
9x + 16|y| & \text{if } 0 < x < |y|, \\
5\sqrt{9x^2 + 16y^2} & \text{if } |y| \leq x.
\end{cases}
\]
(Wolfe)

This was one of the first functions showing that, even in the simple case of a convex function on the 2-dimensional Euclidean Hilbert space, the classical steepest descent algorithm might converge to a point that is different from the unique minimizer and that is not even a critical point. One easily shows that \((-1,0)\) is the unique minimizer with value \( f(-1,0) = -8 \). But steepest descent algorithms often converge to the point \((0,0)\) (cf. [1] [2]) which cannot be a minimizer or critical point by convexity and \( f(-1,0) < f(0,0) \).

\(^2\)There are different implementations of BFGS (in Matlab there is e.g. \texttt{fminunc} and \texttt{E04KAF}, cf. [2]).
We compare various algorithms applied to the Wolfe function in Alt [1, 2] with our results. The starting point is always \((5, 4)\). We apply Algorithm 3.8.A with \(\varepsilon_0 := 0.9\) and we stop as soon as the deviation \(f(x_k) - f(-1, 0)\) is smaller than \(10^{-8}\).

| Algorithm          | BM  | BTR | BFGS | Algorithm 3.8.A |
|--------------------|-----|-----|------|-----------------|
| Iterations         | 37  | 26  | 21   | 16              |
| Gradients          | 37  | 37  | 28   |                 |

\[ f(x_k) - f(-1, 0) \approx 1.4 \cdot 10^{-10} < 10^{-9} < 10^{-8} \approx 2.9 \cdot 10^{-12} \]

One can get even better results with other choices of parameters in (4.1), but for comparability we wanted to keep these parameters fixed for all the benchmark problems. Nevertheless Algorithm 3.8.A always approximates the minimizer well after only a few iterations and gradients also with other parameters (usually 25-35 gradients are needed to get \(f(x_k) - f(-1, 0) < 10^{-8}\)). Thus it seems that we need less or not more iterations and gradients than the bundle methods and comparably many as BFGS.

### 4.2 q-max

Next we consider the q-max function \(f: \mathbb{R}^n \to \mathbb{R}\) given by

\[
  f(x) := \max \left\{ x_i^2 \mid 1 \leq i \leq n \right\} \quad \text{for} \quad x = (x_1, \ldots, x_n) \in \mathbb{R}^n.
\]

\[
  (q\text{-max})
\]

It was used in Alt [2] to demonstrate that the bundle method and the bundle trust region method are fast and stable algorithms for nonsmooth convex functions. Both algorithms have been applied to the three starting points

\[
  u_+ := (1, 2, 3, \ldots, n)
\]

\[
  v := 0.1 \cdot u_+,
\]

\[
  u_\pm := (u_{\pm, 1}, u_{\pm, 2}, \ldots, u_{\pm, n})
\]

where \(u_{\pm, i} := i\) for \(i \leq \frac{n}{2}\) and \(u_{\pm, i} := -i\) otherwise. (In [2] also starting point \(e := (1, \ldots, 1)\) was studied. Since \(\partial f(e)\) is not single-valued and \(\lambda e \in \partial f(e)\) for some \(\lambda \in \mathbb{R}\), an exact linesearch in direction \(-\frac{e}{\|e\|}\) would directly find the global minimizer and, therefore, making the minimization trivial. Unfortunately we do not know which gradients were chosen in [2].)

We apply Algorithm 3.8.A with \(\varepsilon_0 = 0.5\) and the special choice \(T_1(x) = 15 \frac{x}{\varepsilon_0}\). We stop the line search at the first point where the function is not decreasing. The results for the bundle and the bundle trust region method presented for comparison are taken from [2].

A simple argument shows that we get essentially the same iteration points for all starting points \(u_+, u_\pm\) and \(v\), except for scaling with 0.1 in the case of \(v\) and changing sign for the second half of the vectors in the case of \(u_\pm\). Therefore the values of \(f\) at the iteration points are the same, except for multiplying by 0.01 in the case of \(v\). We refrain from giving a proof for that, since we observed this behavior in our numerical computations too.

---

\(3\)We approximate this point numerically and do not compute it analytically.
Algorithm BM Algorithm 3.8.A
Initial point \( u^+ \) \( u_\pm \) \( u^+, u_\pm \) \( v \)
Iterations 142
Gradients 247 199
Value \( 1.090 \cdot 10^{-9} \) \( 4.145 \cdot 10^{-9} \) \( 1.4 \cdot 10^{-10} \) \( 1.4 \cdot 10^{-12} \)

Algorithm BM BTR Algorithm 3.8.A
Initial point \( u^+ \) \( v \) \( u^+ \) \( v \) \( u^+, u_\pm \) \( v \)
Iterations 126
Gradients 3,108 3,140 451 321
Value 95.11 0.01316 1.3 \cdot 10^{-7} 9.6 \cdot 10^{-7} 9.6 \cdot 10^{-6} 9.6 \cdot 10^{-8}

More iterations of Algorithm 3.8.A for \( n = 50 \) and initial points \( u^+, u_\pm \) give
- value \( 1.9 \cdot 10^{-9} \) after 175 iterations and 452 gradients and
- value \( 2.0 \cdot 10^{-11} \) after 200 iterations and 537 gradients
(for initial point \( v \) we get the values \( 1.9 \cdot 10^{-11} \) and \( 2.0 \cdot 10^{-13} \), respectively). Unfortunately we did not find results with other starting points for further comparisons.

The Newton method is just the steepest descent method with proper step size. Thus the step size strategy is crucial, as we can also see from the fact that the bundle trust region method gives much better results than the bundle method. In particular one could expect that Algorithm 3.8.B terminates at the minimizer after at most \( n \) steps for every starting point and properly chosen gradients. In practice we applied Algorithm 3.8.B with \( \sigma_k := \varepsilon_k \) and \( A_k := \text{id} \) as approximation of the Hessian (i.e. we always took the Euclidian norm \( ||\cdot||_k = ||\cdot|| \) and it really stopped exactly at the minimizer after \( n \) steps and \( n \) gradient computations for all 3 initial points and both cases \( n = 20 \) and \( n = 50 \). Notice that the Hessian of \( f \) is typically not regular or it is even not defined. Since we require \( A_k \) to be regular, we set always \( A_k = \text{id} \) which is in fact a (scaled) mean value of all possible Hessians. The crucial difference to Algorithm 3.8.A is the choice of the precise Newton step size \( \varepsilon_{k,0} := ||a_k|| \).

Summarizing we can say that Algorithm 3.8.A gives a good approximation of the minimizer after relatively few iterations and gradient computations. Again it appears that both versions of Algorithm 3.8 are faster than the bundle methods. Due to the essentially quadratic structure of the function it is not surprising that our basic algorithm is inferior to the Newton method (and to BFGS giving the same results). But we can easily exploit the quadratic structure in Algorithm 3.8.B by choosing the parameters properly and obtain the exact solution after only \( n \) steps too.

4.3 Rosenbrock

As next classical benchmark function we consider the Rosenbrock function \( f : \mathbb{R}^2 \rightarrow \mathbb{R} \) given by
\[
f(x, y) = (1 - x)^2 + 100(y - x^2)^2. \tag{Ros}
\]
Here a steepest descent method leads into a canyon and then follows it, which is very time consuming. But the Newton method and Newton based methods like BFGS show a very good performance for this function. The minimizer is \((1, 1)\) with value \( f(1, 1) = 0 \).

In Alt [1] two versions of BFGS, the conjugated gradient method (CG) and two versions of trust region methods (TRM) are applied to the Rosenbrock function where always initial point \((-1.9, 2.0)\) is used. We used the same initial point and applied both Algorithm 3.8.A (with \( \varepsilon_0 := 1.5 \)) and Algorithm 3.8.B (where \( A_k \) is the Hessian at iteration point \( x_k \)). The following table compares the results from [1 Section 4.10.4] with our simulations.
Algorithm BFGS 1 BFGS 2 CG TRM 1 TRM 2

| Algorithm | BFGS 1 | BFGS 2 | CG | TRM 1 | TRM 2 |
|-----------|--------|--------|----|-------|-------|
| Iterations| 24     | 47     | 47 | 31    | 34    |
| Gradients | 25     | 49     | 69 | 93    | 33    |
| Value     | $1.85 \cdot 10^{-6}$ | $< 10^{-5}$ | $< 10^{-3}$ | $3.18 \cdot 10^{-9}$ | $1.3 \cdot 10^{-15}$ |

Algorithm 3.8.A Algorithm 3.8.B

| Algorithm 3.8.A | Algorithm 3.8.B |
|-----------------|-----------------|
| Iterations      | 14              | 19              |
| Gradients       | 29              | 37              |
| Value           | $1.74 \cdot 10^{-6}$ | $2.25 \cdot 10^{-9}$ |

Algorithm 3.8.B is impressive by giving the exact solution $(1, 1)$ after merely 12 iterations (i.e. 12 computations of the Hessian) and 20 gradient computations.

4.4 Hilbert function

For the comparison of Algorithm 3.8.A with a Newton-like algorithm (BFGS), we consider the quadratic Hilbert function $f: \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = x^T \cdot A \cdot x \quad \text{with} \quad A = \left( \frac{1}{i + j - 1} \right)_{1 \leq i, j \leq n} \in \mathbb{R}^{n \times n}. \quad \text{(Hilbert)}$$

It is strictly convex with the unique minimizer $x = 0$. The so-called Hilbert matrix $A$ is very ill-conditioned. In practice, however, rounding errors in the representation of $A$ seem to improve the actual condition (cf. [28]). For quasi-Newton methods, $f$ is perfect by its quadratic form, but bad by its condition.

In Spedicato [28] the BFGS algorithm was applied with starting point

$$x_0 = \left( \frac{4}{1}, \frac{4}{2}, \frac{4}{3}, \ldots, \frac{4}{n} \right)$$

and dimensions

$$n = 10, \quad n = 40 \quad \text{and} \quad n = 80.$$  

We used Algorithm 3.8.A with $\varepsilon_0 = \sqrt{n}$ and the Euclidean norm $\| \cdot \|$. The results are given in the following table.

| n  | 10 | 40 | 80 |
|----|----|----|----|
| Iterations | 13 | 43 | 83 |
| Value      | $4 \cdot 10^{-11}$ | $5 \cdot 10^{-9}$ | $1 \cdot 10^{-10}$ |

| Algorithm 3.8.A |
|-----------------|
| n  | 10 | 40 | 80 |
| Iterations | 40 | 40 | 40 |
| Gradients   | 58 | 69 | 77 |
| Value       | $7.0 \cdot 10^{-10}$ | $2.2 \cdot 10^{-10}$ | $3.8 \cdot 10^{-10}$ |

The (smooth) Newton method applied to the Hilbert function would end up in the minimizer after just one step in the case of exact computation. But this does not happen in practice and,
thus, the results depend on the solving algorithm for linear equations. This aspect has been studied intensively for this problem and is beyond the scope of this paper. Therefore we do not apply Algorithm 3.8.B that would give again the (smooth) exact Newton method.

Since the BFGS algorithm is designed for quadratic like functions, we did not expect these relatively good results of our computations (depending, of course, on the initial radius \( \varepsilon_0 \)). It seems that our algorithm can compensate the bad condition of \( A \) by the treatment of neighborhoods that gives a certain robustness even for smooth functions. Let us still mention that first simulations with Algorithm 3.8 for contact problems, where the stiffness matrix of the strain energy is ill conditioned, are quite promising.

4.5 Nesterov’s Chebyshev-Rosenbrock functions

As in Lewis & Overton [19] we consider the function \( \tilde{f} : \mathbb{R}^n \to \mathbb{R} \) suggested by Nesterov

\[
\tilde{f}(x) := \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_{i+1} - 2x_i^2 + 1)^2
\]

and the nonsmooth version

\[
\hat{f}(x) := \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1} |x_{i+1} - 2x_i^2 + 1|
\]

that are used there to test the BFGS algorithm. Obviously both functions have the unique minimizer \( \overline{x} = (1, \ldots, 1) \) with \( \tilde{f}(\overline{x}) = \hat{f}(\overline{x}) = 0 \). According to [19] the BFGS algorithm typically generates iteration points that rapidly approach the highly oscillating manifold

\[
M := \{ x \in \mathbb{R}^n \mid x_{i+1} = 2x_i^2 - 1, \ i = 1, \ldots, n - 1 \}
\]

and then roughly follows it to the minimizer \( \overline{x} \in M \). Therefore it is reasonable for tests to start near \( M \) and, in particular, the starting point

\[
\hat{x} := (-1, 1, 1, \ldots, 1) \in M
\]

is usually considered. More precisely, BFGS always uses \( \hat{x} \) in the smooth case and takes random starting points for the nonsmooth \( \hat{f} \). We used \( \hat{x} \) in the nonsmooth case and a small perturbation of it in the smooth case.

For the \( i \)-th Chebyshev polynomial \( T_i \) one has \( T_2 \circ T_i = T_{2i} \) and \( T_2(s) = 2s^2 - 1 \). Therefore we get for any \( x = (x_1, x_2, \ldots, x_n) \in M \) with \( x_1 \in [-1, 1] \) that

\[
x_{i+1} = T_{2i}(x_1) = \cos \left( 2^i \arccos(x_1) \right)
\]

where the representation by trigonometric functions can be found in [7]. Thus

\[
M \cap [-1, 1]^n = \{ (x_1, \cos(2^1 \arccos(x_1)), \ldots, \cos(2^{n-1} \arccos(x_1))) \mid x_1 \in [-1, 1] \}
\]

Notice that the \( i \)-th Chebyshev polynomial \( T_i \) oscillates \( 2^i - 1 \) times between values \(-1\) and \(1\), i.e. it reaches \( 2^i - 1 \) times both values \(-1\) and \(1\). Normally all algorithms (ours and others) do not follow the entire manifold \( M \), but skip some of the oscillations. The amount of skipped oscillations essentially depends not only on the choice of parameters (like e.g. \( \varepsilon_0 \)) but also on the used computer. Naturally this stochastic noise increases with the dimension of \( X \) and the related strong increase of oscillations. Therefore this benchmark problem is not recommended to compare the speed of algorithms (cf. [19]).
4.5.1 Smooth version

First we consider the smooth function $\tilde{f}$ where we compare results of the BFGS algorithm taken form [19] with our computations. For Algorithm 3.8.A we use the Euclidean norm, $\varepsilon_0 := 0.5$, and the special choice $T_1(x) = 0.001 \frac{x}{\|x\|}$. Algorithm 3.8.B is applied with $\varepsilon_0 = 0.5$ and $A_k$ being the Hessian at iteration point $x_k$. Moreover we take the slightly perturbed initial point $(-1.05, 1, \ldots, 1)$ for Algorithm 3.8.B, since it finishes after just one step at the minimizer for initial point $(-1,1,\ldots,1)$. The next table presents the results.

| $\tilde{f}$ | BFGS | Algorithm 3.8.A | Algorithm 3.8.B |
|-------------|------|-----------------|-----------------|
| Dimension   | $n = 8$ | $n = 10$ | $n = 8$ | $n = 10$ |
| Iterations  | $\approx 6,700$ | $\approx 50,000$ | $16,683$ | $223,639$ |
| Gradients   | $124,040$ | $1,773,929$ | $4,109$ | $31,600$ |
| Value       | $< 10^{-15}$ | $< 10^{-15}$ | $4 \cdot 10^{-26}$ | $6.4 \cdot 10^{-16}$ |

Let us still mention that Algorithm 3.8.B with the quite large radius $\varepsilon_0 = 15$ for $n = 10$ needs merely 19 iterations and 27 gradients to get an iteration point with value less than $10^{-15}$. Here the iterations jump over a big area where $M$ highly oscillates and, this way, avoid the most difficult part of following $M$. In fact we had to adapt parameters carefully in order to reach that our algorithm follows the oscillations of $M$. Summarizing it turns out that our algorithm is robust enough to follow $M$, but it can also skip an awful region by working with appropriate resultants on a sufficiently large ball. In any case it finds the minimizer, while normally Algorithm 3.8.B is significantly faster than Algorithm 3.8.A. We do not know how long the BFGS algorithm typically followed the oscillations of $M$ and how much was skipped.

4.5.2 Nonsmooth version

Now we consider the nonsmooth function $\hat{f}$. In [19] the BFGS algorithm is applied to $\hat{f}$ with random initial point. For $n = 3$ it produces iteration points with value less than $10^{-8}$, but for $n = 4$ it usually breaks down. Algorithm 3.8.A with $\varepsilon_0 = 0.5$ and starting point $(-1,1,\ldots,1)$ succeeds up to dimension $n = 6$ and is overburdened from $n = 7$.

| $\hat{f}$ | Algorithm 3.8.A |
|-------------|-----------------|
| Dimension   | $n = 3$ | $n = 4$ | $n = 5$ | $n = 6$ |
| Iterations  | $4,365$ | $25,766$ | $219,886$ | $> 1,500,000$ |
| Gradients   | $13,691$ | $106,714$ | $1,124,623$ | |
| Value       | $2.6 \cdot 10^{-15}$ | $3.8 \cdot 10^{-10}$ | $1.0 \cdot 10^{-8}$ | $< 10^{-9}$ |

Notice that the value of $\hat{f}$ near the minimizer is merely the modulus of small numbers instead of its square for $f$. This tells us that the approximation of the minimizers should be of comparable quality in both cases, though the values are larger in the nonsmooth case.

4.5.3 Approximating a critical point which is not a minimizer

Let us still take a short look at the nonsmooth variant $f : \mathbb{R}^n \to \mathbb{R}$ with

$$f(x) = \frac{1}{4}|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1|$$

21
having the same minimizer $\mathbf{x} = (1, \ldots, 1)$ with value $f(\mathbf{x}) = 0$. The BFGS method, the gradient sampling algorithm, and Algorithm 3.8 try to follow the set

$$N = \{ x \in \mathbb{R}^n \mid x_{i+1} = 2|x_i| - 1 : i = 1, \ldots, n-1 \} ,$$

which is, in contrast to $M$, not a differentiable manifold. There are also points on $N$ where $f$ is not differentiable. In particular for $n = 2$, function $f$ is not differentiable at $(0, -1)$ and one has $0 \in \partial f(0, -1)$ though it is not a minimizer. The gradient sampling algorithm and the BFGS algorithm converge to this point for many starting points according to [19]. For Algorithm 3.8.A we observed that it always converges to either the critical point $(0, -1)$ (e.g. for initial point $(-1, 1)$) or the global minimizer $(1, 1)$.

Summarizing all cases, our algorithm appears to be quite robust in the presence of high oscillations.

### 4.6 Chebyshev approximation by exponential sums

The gradient sampling algorithm (GS), which is also based on the concept of generalized gradients on sets, was widely tested in Burke, Lewis & Overton [5]. Let us take the Chebyshev approximation by exponential sums as the most simple problem investigated in [5], but which keeps computations comparable. For a given function $u : [1, 10] \to \mathbb{R}$ we are looking for minimizers $(a, b) \in \mathbb{R}^m \times \mathbb{R}^m$ of the function $\bar{f} : \mathbb{R}^n \to \mathbb{R}$ $(n = 2m)$ given by

$$\bar{f}(a, b) := \left\| u(\cdot) - \sum_{j=1}^{m} a_j e^{-b_j \cdot} \right\|_\infty$$

(where $\| \cdot \|_\infty$ denotes the supremum norm and $b_j(\cdot)$ the product of $b_j$ with the argument).

As in [5] we first discretize the problem by fixing equidistant grid points $t_i$ in $[1, 10]$ with

$$t_i = 1 + 9\frac{i}{N} \quad \text{for} \quad 0 \leq i \leq N := 2000$$

and, thus, we are looking for minimizers $(a, b) \in \mathbb{R}^n$ of

$$f(a, b) := \max_{0 \leq i \leq N} \left\{ u(t_i) - \sum_{j=1}^{m} a_j e^{-b_j t_i} \right\} .$$

Let us mention that a slightly modified problem was solved in [5] to find the minimizer of $\bar{f}$. However it appeared that the minimization of $\bar{f}$ is sufficient and we thus disregarded a further adaption.

Now we fix $u(t) = \frac{1}{t}$ on $[1, 10]$ such that, with the smooth functions

$$h_i(a, b) := \frac{1}{t_i} - \sum_{j=1}^{m} a_j e^{-b_j t_i} ,$$

we have to minimize

$$f(a, b) := \max_{0 \leq i \leq N} \max \{ h_i(a, b), -h_i(a, b) \} \quad (4.3)$$

on $\mathbb{R}^n$.

First we present the results from [5] obtained with the gradient sampling algorithm. Here the initial point $a = b = 0$ is used and $2n$ gradients had to be computed in each iteration. Due to lack of precision the algorithm didn’t succeed for $n > 8$.

---

4 We recall that the gradient sampling algorithm needs at least $n + 1$ gradients in each iteration.
For the application of Algorithm 3.8, we start with some observations. If we interpret the graph of $f$ as the surface of some mountains, the initial point $a = b = 0$ is located at some ridge where the path of steepest descent follows that ridge until a saddle point. This means for the algorithm that all iteration points $(a_k, b_k)$ and the corresponding gradients $f'_k = (a'_k, b'_k)$ have the form

$$a_{k,j} = a_{k,1}, \quad b_{k,j} = b_{k,1}, \quad a'_{k,j} = a'_{k,1}, \quad b'_{k,j} = b'_{k,1} \quad \text{for all } j = 1, \ldots, m$$

(which can be shown analytically by an easy induction argument) and the sequence $(a_k, b_k)$ converges to a saddle point $(\hat{a}, \hat{b})$ with

$$\hat{a} = \frac{2}{n} (a, \hat{a}, \ldots, \hat{a}) \in \mathbb{R}^m, \quad \hat{b} = \frac{2}{n} (\hat{b}, \hat{b}, \ldots, \hat{b}) \in \mathbb{R}^m$$

(e.g. $(\hat{a}, \hat{b}) \approx (1.43, 0.45) \in \mathbb{R}^2$ for $n = 2$). It is remarkable that our algorithm also in practice precisely follows that ridge and ends up in the saddle point. This however means that we need some slight perturbation to find the minimizer. Therefore we consider both a perturbed initial point

$$a = -0.001 \cdot (0^2, 2^2, \ldots, (n - 2)^2) \quad \text{and} \quad b = 0.001 \cdot (1^2, 3^2, \ldots, (n - 1)^2)$$

and a perturbed function

$$\hat{f}(a, b) = f(a, B b) \quad \text{for some regular } B \in \mathbb{R}^{m \times m}. $$

More precisely, we replace $h_i(a, b)$ in (4.3) with

$$\hat{h}_i(a, b) := \frac{1}{t_i} - \sum_{j=1}^m a_j e^{-jb_j t_i}$$

(i.e. $b_j$ is substituted by $jb_j$) and minimize

$$\hat{f}(a, b) := \max_{0 \leq j \leq N} \max \left\{ \hat{h}_j(a, b), -\hat{h}_j(a, b) \right\}. \quad (4.4)$$

Notice that this change of $f$ does not effect the minimal value and, thus, our results can be compared with those of the gradient sampling algorithm. The next table presents the results of Algorithm 3.8 with $\varepsilon_0 = 5\sqrt{m}$ and the special choice $T_2(x) = 0.1 \cdot x$.
The deliberate choice of gradients might be some reason that we need much less gradients than the gradient sampling method. We can also handle higher dimensions and reach our computational limit due to rounding errors for \( n = 14 \).

### 4.7 Nonlinear regression

We finally consider a nonlinear regression problem that can be found in Alt [1, Section 2.3.1]. More precisely, we want to minimize \( f : \mathbb{R}^3 \to \mathbb{R} \) given by

\[
 f(x) = \sum_{i=1}^{10} \left( x_1 e^{x_2} + x_3 - \eta_i \right)^2
\]

for values

\[
\begin{array}{ccccccccccc}
 i \\
 \eta_i \\
 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
 1.0 & 1.1 & 1.2 & 1.35 & 1.55 & 1.75 & 2.5 & 3.0 & 3.7 & 4.5 \\
\end{array}
\]

It turns out that the steepest descent method, the Nelder-Mead method, and the BFGS method are only partially able to solve this problem for the initial points \( x_0 = (0, 0, 0) \) and \( x_0 = (1, 1, 1) \). More precisely we get from [1] that:

- The steepest descent method stops for both initial points at some useless point (i.e. it is far from being a minimizer).
- For initial point \( x_0 = (1, 1, 1) \), the BFGS algorithm and the Nelder-Mead method also stop at some useless point. With initial point \( x_0 = (0, 0, 0) \), the BFGS algorithm reaches a point with value 0.0861942 after 21 iterations and the Nelder-Mead method is also successful with a slightly larger value.

We successfully applied Algorithm 3.8.A and Algorithm 3.8.B with \( A_k \) being the Hessian at iteration point \( x_k \) and with \( \varepsilon_0 = 0.5 \) to both initial points.

| Algorithm   | Algorithm 3.8.A | Algorithm 3.8.B |
|-------------|-----------------|-----------------|
| Initial point | \( (0, 0, 0) \) | \( (0, 0, 0) \) |
| Iterations   | 56              | 70              |
| Gradients    | 130             | 194             |
| Value        | 0.0861942       | 0.0861942       |

We also tested Algorithm 3.8 with other starting points and always obtained the same minimal value of \( f \) and the same minimizer which is approximately

\[(0.270, 0.269, 0.592)\,.

Though Algorithm 3.8 is slower than the BFGS algorithm for the first initial point, our algorithm seems to be much more robust in the sense that it always finds the minimal solution (notice that the actual minimizer is unknown). It is interesting to see that even in this smooth case Algorithm 3.8.A shows better convergence than Algorithm 3.8.B with variable norms.

### 4.8 Summary

Summarizing we can say that Algorithm 3.8 found the minimizer not only in all cases where the others succeeded, but also in cases where others failed. Moreover, for many problems our new algorithm needed the least iterations and gradient computations, i.e. it was the fastest in
this sense. In particular in higher dimensions the potential of Algorithm 3.8 became clearly visible.

It turns out that Algorithm 3.8 is certainly an alternative for the solution of nonlinear and especially nonsmooth minimization problems. Stability and robustness of the algorithm are very convincing. It not only avoids typical oscillations of classical smooth schemes, it also follows highly oscillating descent paths of very nasty functions and it can precisely trace some mountain ridge.

Let us mention that we did not yet exploit the full potential of the algorithm, since e.g. the choices of $G$, $T_1$, $T_2$ and the step size control based on $\varepsilon_{k,i}$ are not yet optimized by a systematic investigation.\footnote{For many of the considered benchmark problems slower decreasing functions like $T_2(x) = K\sqrt{x}$ instead of $T_2(x) = Kx$ lead to better results for large dimensions. Using $G(x, y) > y$ we also got better results in many cases.} Also efficient stopping criteria are not yet considered. Nevertheless the achieved results are a promising basis for the treatment of relevant variational problems. Simulations for the highly degenerate eigenvalue problem of the 1-Laplacian will be presented in an upcoming paper.

References

[1] W. Alt. *Nichtlineare Optimierung*. Vieweg, Wiesbaden 2002.
[2] W. Alt. *Numerische Verfahren der konvexen, nichtglatten Optimierung*. Teubner, Wiesbaden 2004.
[3] A. M. Bagirov, B. Karasözen, M. Sezer. Discrete gradient method: A derivative free method for nonsmooth optimization. J. Optim. Theory Appl. 137 (2008) 317-334.
[4] A. Beck. *First-Order Methods in Optimization*. SIAM, Philadelphia 2017.
[5] J. V. Burke, A. S. Lewis, M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. 15 (2005) 751-779.
[6] A. Chambolle, T. Pock. A first-order primal-dual algorithm for convex problems with applications in imaging. J. Math. Imaging Vision 40 (2011) 120-145.
[7] E. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, New York 1966.
[8] F. H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley, New York 1983.
[9] J. Eckstein, D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. 55 (1992) 293-318.
[10] A. Frangioni. Generalized bundle methods. SIAM J. Optim. 13 (2002) 117-156.
[11] M. Gaudioso, M. F. Monaco. A bundle type approach to the unconstrained minimization of convex, nonsmooth functions. Math. Program. 23 (1982) 216-226.
[12] A. A. Goldstein. Cauchy’s method of minimization. Numer. Math. 4 (1962) 146-150.
[13] A. A. Goldstein. On steepest descent. J. SIAM Control Ser. A 3 (1965) 147-151.
[14] A. A. Goldstein. Optimization of Lipschitz continuous functions. Math. Program. 13 (1977) 14-22.
[15] L. Hörmander. Sur la fonction d’appui des ensembles convexes dans un espace localement convexe. Ark. Mat. 3 (1955) 181-186.
[16] J. B. Hiriart-Urruty, C. Lemarechal. *Convex Analysis and Minimization Algorithms*, Vol. 1&2. Springer, New York 1993.
[17] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Springer, Berlin 1985.
[18] K. C. Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. 18 (2007) 379-388.
[19] A. S. Lewis, M. L. Overton. Nonsmooth optimization via BFGS. \url{http://www.cs.nyu.edu/overton/papers/pdffiles/bfgs_inexactLS.pdf} (2008)
[20] A. Lewis, M. Overton. Nonsmooth optimization via quasi-Newton methods. Math. Program. 141 (2013) 135-163.
[21] M. M. Makela, P. Neittaanmaki. *Nonsmooth Optimization*. World Scientific, Singapore 1992.

[22] J. Mankau, F. Schuricht. Gradients on sets. to appear in J. Convex Anal.

[23] R. Mifflin. An algorithm for constrained optimization with semismooth functions. Math. Oper. Res. 2 (1977) 191-207.

[24] J. Nocedal, S. J. Wright. *Numerical Optimization*. Springer, New York 2006.

[25] E. Polak, J. O. Royset. Algorithms for finite and semi-infinite minmaxmin problems using adaptive smoothing techniques. J. Optim. Theory Appl. 119 (2003) 421-457.

[26] R. Pytlak. *Conjugate Gradient Algorithms in Nonconvex Optimization*. Springer, Berlin 2009.

[27] H. Schramm. *Eine Kombination von Bundle- und Trust-Region-Verfahren zur Lösung nichtdifferenzie- rer Optimierungsprobleme*. Bayreuther Mathematische Schriften (vol. 30), Bayreuth 1989.

[28] E. Spedicato. Computational experience with quasi-newton algorithms for minimization problems of moderately large size. Report CISE-N-175, Milano 1976.

[29] M. Ulbrich. Semismooth Newton methods for operator equations in function spaces. SIAM J. Optim. 13 (2006) 805-841.

[30] D. Werner. *Funktionalanalyis* (5th ed.). Springer, Berlin 2005.

[31] P. H. Wolfe. A method of conjugate subgradients of minimizing nondifferentiable convex functions. Math. Program. 3 (1975) 145-173.

[32] E. Zeidler. *Nonlinear Functional Analysis and its Applications, vol. III: Variational Methods and Optimization*. Springer, New York 1985.

[33] J. Zowe. Nondifferentiable optimization. In: K. Schittkowski (ed). Computational Mathematical Programming. NATO ASI Series (Series F: Computer and Systems Sciences) vol. 15, Springer, Berlin 1985, pp. 323-356.