Numerical infinitesimals in a variable metric method for convex nonsmooth optimization.

Manlio Gaudioso[a], Giovanni Giallombardo[a], Marat Mukhametzhanov[a,b]

[a] *Università della Calabria, Rende, Italy*
[b] *Lobachevsky State University, Nizhni Novgorod, Russia*

Abstract

The objective of the paper is to evaluate the impact of the infinity computing paradigm on practical solution of nonsmooth unconstrained optimization problems, where the objective function is assumed to be convex and not necessarily differentiable. For such family of problems, the occurrence of discontinuities in the derivatives may result in failures of the algorithms suited for smooth problems.

We focus on a family of nonsmooth optimization methods based on a variable metric approach, and we use the infinity computing techniques for numerically dealing with some quantities which can assume values arbitrarily small or large, as a consequence of nonsmoothness. In particular we consider the case, treated in the literature, where the metric is defined via a diagonal matrix with positive entries.

We provide the computational results of our implementation on a set of benchmark test-problems from scientific literature.

Keywords: nonsmooth optimization; infinity computing; variable-metric methods

1. Introduction

Nonsmooth (or nondifferentiable) optimization is about finding a local minimum of a real-valued function of several variables, that is solving the following unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a function not necessarily differentiable.

Although nonsmoothness generally occurs at a zero-measure set of the function domain, convergence to a minimum is not ensured in case an algorithm designed for treating smooth problems is applied to nonsmooth ones.

For the above reasons, intensive research activities have been developed in last decades, and several proposals for dealing with nonsmoothness are offered in the literature. As for historical contributions, we mention here the books [10, 21, 39] and the seminal papers [6, 16].

In this paper we focus on the unconstrained minimization of a convex function under no differentiability hypothesis. In this area two research mainstreams are active: subgradient

Email address: giovanni.giallombardo@unical.it (Giovanni Giallombardo)

type methods (see the classic version in [39] and, among the others, the more recent variants in [4, 12, 30]) and bundle type methods. The latter stems from the seminal paper [23], and benefits from both the cutting plane model [6, 16] and the conjugate subgradient approach [40]. The term *bundle* is referred to a certain amount of information (values and subgradients of the objective functions at a set of points in its domain) which is accumulated and possibly updated as the algorithm proceeds. Bundle methods [15] require, at each iteration, the solution of a linear (or, more often, quadratic) program aimed at finding a tentative displacement from the current approximation of the minimizer. Due to nonsmoothness, even in case a line search is adopted, a sufficient decrease in the objective function is not guaranteed and, consequently, this family of methods accommodates for the so called *null step*, a situation where no progress toward the minimum is achieved and, instead, some additional information about the local behaviour of the objective function is accumulated into the bundle.

Literature on bundle methods is very rich. We cite here the contributions given in [2, 13, 9, 14, 22, 26]. Some authors have designed methods that retain many features of the classic bundle approach, while trying to simplify the displacement finding phase, thus avoiding solution of a too burdensome subproblem at each iteration, see [17, 20, 27]. In addition, such methods utilize some ideas coming from the vast literature of the variable metric approach to smooth optimization, as they embed variants of standard Quasi–Newton updating formulae for approximating the Hessian matrix (see [11] for a classic survey on Quasi–Newton methods and [3], with the references therein, for the applications of the variable metric approach to nonsmooth optimization).

A Quasi–Newton method for minimizing a differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is an iterative method where, at any point $\mathbf{x}^k \in \mathbb{R}^n$, a matrix B^k is generated as an approximation of the Hessian matrix such that it satisfies the equation

$$B^k \mathbf{s}^k = \mathbf{u}^k, \tag{2}$$

where

$$\mathbf{s}^k \triangleq \mathbf{x}^k - \mathbf{x}^{k-1} \tag{3}$$

and

$$\mathbf{u}^k \triangleq \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1}). \tag{4}$$

Sometimes equation (2) is replaced by

$$H^k \mathbf{u}^k = \mathbf{s}^k, \tag{5}$$

where H^k is the inverse of B^k.

Methods extending the Quasi–Newton idea to the convex nondifferentiable case still require at each iteration satisfaction of equation (2), the only difference being in the definition of vector $\mathbf{u}^k$. In fact, denoting by $\partial f(\mathbf{x})$ the subdifferential and by $\mathbf{g} \in \partial f(\mathbf{x})$ any subgradient of f at $\mathbf{x}$, see [15], $\mathbf{u}^k$ can be restated as

$$\mathbf{u}^k \triangleq \mathbf{g}_k - \mathbf{g}_{k-1}, \tag{6}$$

with $\mathbf{g}_k$ and $\mathbf{g}_{k-1}$ being subgradients of f at the points $\mathbf{x}^k$ and $\mathbf{x}^{k-1}$, respectively.

However, in the case of nondifferentiable functions, the search for a matrix B^k satisfying (2) is a problem inherently ill-conditioned, due to possible discontinuities in the first order derivatives, which in turn may result in "large" $\mathbf{u}^k$ corresponding to "small" $\mathbf{s}^k$.

This observation is the main motivation of this paper, as we propose to tackle equation (2) by applying the approach recently proposed in [34, 35] to handle infinite and infinitesimal numbers, and based on the *grossone* concept, a numeral resuming the properties of the entire set of natural number.

We remark that the objective of the paper is not to introduce yet another method for solving convex nondifferentiable minimization problems, but to experiment how ideas coming from the *grossone*-based arithmetic can be cast into a traditional numerical optimization framework.

The paper is organized as follows. Some basic notions about the *grossone* approach are reported in §2. Next, in §3, we introduce the relevant details of a *grossone*-based Quasi-Newton method for nonsmooth optimization. Finally, in §4, the results of our computational experiments on some benchmark test-problems are reported.

2. The numeral system

We embed into the treatment of the optimization problem (1) a recently introduced computational methodology allowing one to work *numerically* with infinities and infinitesimals in a handy way (see the detailed survey [33] and the informative book [36]). This computational methodology has already been successfully applied in optimization and numerical differentiation [8, 42], and in a number of other theoretical and computational research areas such as cellular automata [7], Euclidean and hyperbolic geometry [28], percolation [19], fractals [37], infinite series and the Riemann zeta function [41], the first Hilbert problem, Turing machines, and supertasks [31], numerical solution of ordinary differential equations [1, 29, 38], etc.

This methodology uses a numeral system working with an infinite number called *grossone*, expressed by the numeral ①, and introduced as the number of elements of the set of natural numbers (the non-contradictory nature of the methodology has been studied in [25]). This numeral system allows one to express a variety of numbers involving different infinite and infinitesimal parts, and to execute operations with all of them in a unique framework. Notice that such numerical approach is not related to the well known non-standard analysis framework (see [32]) that has a symbolic character and, therefore, allows symbolic computations only. The *grossone* approach, instead, allows full numerical treatment of both infinite and infinitesimal numbers.

In the new positional system based on the infinite radix ① any number C expressed as

$$C = c_{p_m}①^{p_m} + c_{p_{m-1}}①^{p_{m-1}} + ... + c_{p_1}①^{p_1} + c_{p_0}①^{p_0} + c_{p_{-1}}①^{p_{-1}} + ... + c_{p_{-k}}①^{p_{-k}}, \qquad (7)$$

is written in the form:

$$C = c_{p_m}①^{p_m}...c_{p_1}①^{p_1}c_{p_0}①^{p_0}c_{p_{-1}}①^{p_{-1}}...c_{p_{-k}}①^{p_{-k}}. \qquad (8)$$

Here, all numerals $c_i \neq 0$ belong to a traditional numeral system and are called *grossdigits*, while all numbers p_i, called *grosspowers*, are sorted in decreasing order with $p_0 = 0$:

$$p_m > ... > p_1 > p_0 > p_{-1} > ... > p_{-k}. \qquad (9)$$

The term corresponding to $p_0 = 0$ represents the finite part of the number C, the terms having positive finite or infinite grosspowers represent the infinite parts of C, and the terms having finite or infinite negative grosspowers are the infinitesimal parts of C. So, infinite and infinitesimal numbers can be described in the presented numeral system as well as traditional finite numbers in a unique framework.

3. The algorithm

In order to evaluate the role and the impact of the infinity computing paradigm on the effectiveness of variable-metric methods applied to the nonsmooth problem (1), we have revisited the Diagonal Bundle method (`D-Bundle`) introduced in [20]. `D-Bundle` is based on the limited-memory bundle approach [17], where ideas coming from the variable-metric bundle approach [24, 27] are combined with the extension to nonsmooth problems of the limited-memory approach introduced in [5]. The main feature of the method is the use of the diagonal update formula of the variable-metric matrix introduced in [18]. We remark that `D-Bundle` is suited for dealing with nonconvexity as well as with nonsmoothness. The reader is referred to [20] for a thorough description of the method, along with its convergence properties and several numerical results. We restrict our attention to the convex nonsmooth case, where most of the literature on nonsmooth optimization has concentrated in last decades. This allows us to adopt a simplified structure of `D-Bundle`, thus highlighting the specific impact of the infinity computing paradigm. The relevant details of our `Grossone-D-Bundle` algorithm are presented in the following.

We assume that at each point $\mathbf{x} \in \mathbb{R}^n$ it is possible to calculate $f(\mathbf{x})$ and a subgradient $\mathbf{g} \in \partial f(\mathbf{x})$. Letting $\mathbf{x}^k$ be the estimate of the minimum at iteration k, the search direction $\mathbf{d}^k$ adopted to locate the next iterate is defined as

$$\mathbf{d}^k = -H^k \boldsymbol{\xi}_a^k, \tag{10}$$

where $\boldsymbol{\xi}_a^k$ is the current aggregate subgradient (see the details below), and H^k is the inverse of B^k, the positive definite variable-metric $n \times n$ matrix, resembling the classic approximation of the Hessian matrix adopted in quasi-Newton methods for smooth optimization. We remark that in calculating $\mathbf{d}^k$, unlike standard bundle methods, no solution of any optimization subproblem is required.

Once the search direction $\mathbf{d}^k$ is available, a line search along $\mathbf{d}^k$ is performed, which can return two possible outcomes: *serious* step, whenever a sufficient decrease with respect to the desirable one is achieved, and *null* step otherwise. In the former case, a new approximation $\mathbf{x}^{k+1}$ of a minimizer is obtained, while in the latter an auxiliary point $\mathbf{y}^{k+1}$ is gathered with an associate subgradient, to enrich the information about the local behavior of the function. In fact, the definition of the aggregate subgradient in formula (10) is different in the two cases. If a serious step has been performed (i.e., the new iterate $\mathbf{x}^{k+1}$ has been located) the aggregate subgradient $\boldsymbol{\xi}_a^{k+1}$ is any subgradient of f at $\mathbf{x}^{k+1}$. On the other hand, in the null-step case, no move from the current estimate of the minimizer is made (that is $\mathbf{x}^{k+1} = \mathbf{x}^k$), hence the aggregate subgradient $\boldsymbol{\xi}_a^{k+1}$ is obtained as a convex combination of the three vectors $\mathbf{g}^k \in \partial f(\mathbf{x}^k)$, $\mathbf{g}^{k+1} \in \partial f(\mathbf{y}^{k+1})$, and $\boldsymbol{\xi}_a^k$, with multipliers λ_1^*, λ_2^*, and λ_3^*,

respectively, which minimize the following function

$$\phi(\lambda_1,\lambda_2,\lambda_3) \triangleq \frac{1}{2}\left(\lambda_1\mathbf{g}^k+\lambda_2\mathbf{g}^{k+1}+\lambda_3\boldsymbol{\xi}_a^k\right)^\top H^k\left(\lambda_1\mathbf{g}^k+\lambda_2\mathbf{g}^{k+1}+\lambda_3\boldsymbol{\xi}_a^k\right)+\lambda_2\alpha^{k+1}+\lambda_3\alpha_a^k, \tag{11}$$

where α^{k+1} is the standard (nonnegative) linearization error

$$\alpha^{k+1} \triangleq f(\mathbf{x}^k)-f(\mathbf{y}^{k+1})-(\mathbf{g}^{k+1})^\top(\mathbf{x}^k-\mathbf{y}^{k+1}),$$

and α_a^k is the aggregated linearization error. The aggregated linearization error is updated by the following recursive equality

$$\alpha_a^{k+1}=\lambda_2^*\alpha^{k+1}+\lambda_3^*\alpha_a^k,$$

and it is initialized to zero every time a serious step takes place. We remark that minimization of function (11) can be easily obtained, see [27, Section 4].

Now we focus on the updating technique of matrix B^k. As in [20] matrix B^k must be kept diagonal and positive definite. First we introduce a simplified version of the procedure indicated in [20]. In particular, at iteration k one can only store the information about the previous iterate, and calculate two correction vectors $\mathbf{s}^k$ and $\mathbf{u}^k$, according to the definitions (3) and (6), respectively. Hence, following [20], matrix B^k is obtained by solving the following optimization problem

$$\begin{aligned}
\min \quad & \|B\mathbf{s}^k-\mathbf{u}^k\| && && (12)\\
\text{s.t.} \quad & B_{ii}\geq\epsilon, && \forall i\in\{1,\dots,n\}, && (13)\\
& B_{ij}=0, && \forall i\neq j\in\{1,\dots,n\}, && (14)
\end{aligned}$$

for some $\epsilon>0$, whose optimal solution can be expressed as

$$B_{ii}^k=\max\left(\epsilon,\frac{\mathbf{u}_i^k}{\mathbf{s}_i^k}\right),\ \ i=1,\dots,n. \tag{15}$$

We remark that the nonsmoothness of f is likely to cause ill-conditioning of problem (12)-(14), as the elements of matrix B^k may result arbitrarily large and, consequently, those of H^k arbitrarily small, since

$$H_{ii}^k=(B_{ii}^k)^{-1},\ \ i=1,\dots,n. \tag{16}$$

This is where the infinity computing paradigm comes into play. In order to control ill-conditioning we replace first the correction vectors $\mathbf{s}^k$ and $\mathbf{u}^k$ with vectors $\boldsymbol{\delta}^k$ and $\boldsymbol{\gamma}^k$, respectively, whose components are defined as follows

$$\boldsymbol{\delta}_i^k=\begin{cases}\mathbf{s}_i^k, & \text{if } |\mathbf{s}_i^k|>\epsilon,\\ ①^{-1}, & \text{otherwise,}\end{cases} \tag{17}$$

and

$$\boldsymbol{\gamma}_i^k=\begin{cases}\mathbf{u}_i^k, & \text{if } |\mathbf{u}_i^k|>\epsilon,\\ ①^{-1}, & \text{otherwise.}\end{cases} \tag{18}$$

Then, we possibly correct the ratio $\dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k}$ by introducing the following

$$\mathbf{b}_i^k = \begin{cases} ①^{-1}, & \text{if } 0 < \dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k} \leq \epsilon \\ \dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k}, & \text{otherwise.} \end{cases} \tag{19}$$

Finally, by analogy with the solution of problem (12)-(14), we set the elements of the diagonal matrix B^k according to the rule

$$B_{ii}^k = \max\left(①^{-1}, \mathbf{b}_i^k\right), \quad i = 1, \ldots, n. \tag{20}$$

Note that matrix B^k may contain infinite and infinitesimal numbers. However, since calculation of $\mathbf{d}^k$ according to formula (10) has to take place in a classical arithmetic framework, we need to get rid of the dependence on ① and $①^{-1}$ in the definition of matrix H^k. Therefore we choose the following scheme which, as far as finite numbers are involved, reflects the standard inverse calculation scheme:

$$H_{ii}^k = \begin{cases} (B_{ii}^k)^{-1} & \text{if } B_{ii}^k \text{ neither depends on ① nor on } ①^{-1}, \\ (B_{ii}^k)^{-1} \cdot ① & \text{if } B_{ii}^k \text{ is of the type } \alpha①, \ \alpha \in \mathbb{R} \\ (B_{ii}^k)^{-1} \cdot ①^{-1} & \text{if } B_{ii}^k \text{ is of the type } \alpha①^{-1}, \ \alpha \in \mathbb{R} \end{cases} \tag{21}$$

A better understanding of the consequences of (21) can be gained by examining the following example.

Example 3.1. Let the vectors

$$\mathbf{s}^\top = \begin{pmatrix} 1.0 \times 10^{-4} & 1.0 \times 10^{-6} & 1.0 \times 10^{-4} \end{pmatrix}$$

and

$$\mathbf{u}^\top = \begin{pmatrix} -1.0 \times 10^{-4} & 2.0 \times 10^{1} & 1.0 \times 10^{-5} \end{pmatrix}$$

be given, where we have dropped the superscript k for notational simplicity. For each value of $\epsilon \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, we calculate the matrices B_ϵ and H_ϵ according to (15) and (16), and the matrices $B_①$ and $H_①$ according to (20) and (21), and we report the results in Figures 1, 2, and 3, respectively. The effect of adopting the rules (17)–(21) can be seen by comparing $H_①$ against H_ϵ, since $H_①$ "stabilizes" as we take smaller and smaller values of ϵ. In fact, one can easily observe that taking smaller values than 10^{-8} for ϵ, then the ill-conditioning of B_ϵ gets worse and worse, while $H_①$ remains unchanged, see Figure 3.

Now we are ready to introduce the formal statement of our `Grossone-D-Bundle` method, reported in Algorithm 1. The following parameters need to be set: the sufficient decrease parameter $m \in (0, 1)$, the matrix updating threshold $\epsilon > 0$, the stepsize reduction parameter $\sigma \in (0, 1)$, the stopping parameter $\eta > 0$, and the null step parameter $\theta > 0$. Few comments on the algorithm mechanism are in order. At Step 4, a search direction $\mathbf{d}^k$ at the current point $\mathbf{x}^k$ is selected according to (10). Next, at Step 5, the desirable reduction w^k of the

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-3} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \qquad B_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 2.0 \times 10^{1}① & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{3} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \qquad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-2} & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

Figure 1: Variable-metric matrices with $\epsilon = 10^{-3}$

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-5} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \qquad B_{①} = \begin{pmatrix} ①^{-1} & 0 & 0 \\ 0 & 2.0 \times 10^{1}① & 0 \\ 0 & 0 & ①^{-1} \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{5} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \qquad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-2} & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

Figure 2: Variable-metric matrices with $\epsilon = 10^{-5}$

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-8} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \qquad B_{①} = \begin{pmatrix} ①^{-1} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{8} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \qquad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix}$$

Figure 3: Variable-metric matrices with $\epsilon = 10^{-8}$

objective function is calculated, and the algorithm stops at Step 6 if w^k is very close to zero. Otherwise, a line-search procedure along $\mathbf{d}^k$ is started at Step 9, whose termination depends on fulfillment of the sufficient decrease condition at Step 10. In such a case, a serious step takes place along with the *grossone*-based update of matrix H^k. If no sufficient decrease is attained at Step 10, then the line-search is iterated at Step 21, unless the step size has become very small, in that case a null step occurs, which amounts to updating the aggregate gradient $\boldsymbol{\xi}_a^k$ and the linearization error α_a^k, but not the matrix H^k. For further details, especially regarding the definition of the desirable descent w^k, and the calculation of $\boldsymbol{\xi}_a^k$ and α_a^k in the null-step case, we refer the reader to [20].

Convergence properties of the proposed algorithm are consequence of those described in [20], where it is required that no update of matrix H_k takes place in case of null step (this is what actually occurs in our implementation too), and that matrix H_k stays bounded. In our case, boundedness can be easily checked by enumerating all possible cases in the construction of matrix H_k, taking into account, in particular, the definition (19).

Algorithm 1 `Grossone-D-Bundle`

Input: a starting point $\mathbf{x}^0 \in \mathbb{R}^n$, parameters $\theta > 0$, $\eta > 0$, $\epsilon > 0$, $m \in (0,1)$, $\sigma \in (0,1)$
Output: an approximate local minimizer $\mathbf{x}^* \in \mathbb{R}^n$

1: Calculate $\mathbf{g}^0 \in \partial f(\mathbf{x}^0)$, and set $\boldsymbol{\xi}_a^0 = \mathbf{g}^0$ ▷ Initialization
2: Set $H^0 = I$ (i.e., the $n \times n$ identity matrix) ▷ |
3: Set $\alpha^0 = \alpha_a^0 = 0$, and $k = 0$ ▷ |
4: Set $\mathbf{d}^k = -H^k \boldsymbol{\xi}_a^k$ ▷ Find a search direction at $\mathbf{x}^k$
5: Set $w^k = (\boldsymbol{\xi}_a^k)^\top \mathbf{d}^k - 2\alpha_a^k$ ▷ Set the desirable reduction
6: **if** $w^k \geq -\eta$ **then** ▷ Stopping test
7: set $\mathbf{x}^* = \mathbf{x}^k$ and **exit** ▷ Return $\mathbf{x}^*$ as an approximate minimizer
8: **end if**
9: Set $t_1 = 1$ and $s = 1$ ▷ Start the line-search
10: **if** $f(\mathbf{x}^k + t_s \mathbf{d}^k) \leq f(\mathbf{x}^k) + m t_s w^k$ **then** ▷ Descent test
11: Set $\mathbf{x}^{k+1} = \mathbf{x}^k + t_s \mathbf{d}^k$ ▷ Make a serious step
12: Calculate $\mathbf{g}^{k+1} \in \partial f(\mathbf{x}^{k+1})$ ▷ |
13: Set $\boldsymbol{\xi}_a^{k+1} = \mathbf{g}^{k+1}$ ▷ |
14: Calculate H^{k+1} according to (21) ▷ Grossone matrix update
15: Set $k = k + 1$ and **go to 4**
16: **end if**
17: **if** $t_s \leq \theta$ **then** ▷ Closeness test
18: Calculate $\mathbf{g}_+ \in \partial f(\mathbf{x}^k + t_s \mathbf{d}^k)$ and $\boldsymbol{\xi}_a \in \mathrm{conv}\{\mathbf{g}^k, \boldsymbol{\xi}_a^k, \mathbf{g}_+\}$ ▷ Make a null step
19: Set $\alpha_+ = f(\mathbf{x}^k) - f(\mathbf{x}^k + t_s \mathbf{d}^k) + t_s g_+^\top \mathbf{d}^k$ ▷ |
20: Calculate $\alpha_a \in \mathrm{conv}\{0, \alpha_a^k, \alpha_+\}$ ▷ |
21: Update $\boldsymbol{\xi}_a^k = \boldsymbol{\xi}_a$, $\alpha_a^k = \alpha_a$, and **go to 4** ▷ |
22: **else**
23: Set $t_{s+1} = \sigma t_s$, $s = s + 1$ and **go to 10** ▷ Iterate the line-search
24: **end if**

4. Numerical results

Numerical experiments have been executed on three different classes of large-scale test problems taken from [3] and reported in Table 1.

Chained LQ

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{-x_i - x_{i+1}, -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\right\}$$

$\mathbf{x}_i^0 = -0.5$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1/\sqrt{2}$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = -(n-1)\sqrt{2}$

Chained CB3 I

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{x_i^4 + x_{i+1}^2, (2 - x_i)^2 + (2 - x_{i+1})^2, 2e^{-x_i + x_{i+1}}\right\}$$

$\mathbf{x}_i^0 = 2$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = 2(n-1)$

Chained CB3 II

$$f(\mathbf{x}) = \max\left\{\sum_{i=1}^{n-1}(x_i^4 + x_{i+1}^2), \sum_{i=1}^{n-1}\left((2 - x_i)^2 + (2 - x_{i+1})^2\right), \sum_{i=1}^{n-1}(2e^{-x_i + x_{i+1}})\right\}$$

$\mathbf{x}_i^0 = 2$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = 2(n-1)$

Table 1: Test problems ($\mathbf{x}^0$ is the starting point, $\mathbf{x}^*$ is the minimizer)

We have adopted different values of the space dimension n, running our experiments for $n = 50,\ 100,\ 200$. The algorithm has been coded in C++, compiled with Microsoft Visual Studio 2010 Professional on a HP-15-ba090ur machine, under the MS Windows 10 operating system. We report the results obtained by stopping the algorithm after a given number N_f of function evaluations, where $N_f \in \{50, 100, 200, 300, 400, 500\}$. In particular, we provide the objective function value f^*, the relative error $e_r = \frac{|f^* - f(x^*)|}{1 + |f(x^*)|}$, the number of serious steps N_S, and the number of H^k updates that involved the use of grossone $N_{①}$. We have tested the algorithm for several values of ϵ and, for simplicity of presentation, we only report the results obtained adopting the two extreme values $\epsilon = 10^{-2}$, see Tables 2 to 4, and $\epsilon = 10^{-10}$,

see Tables 5 to 7. The remaining parameters of the algorithms have been set as follows: $\sigma = 0.7$, $m = 0.1$, $\eta = 10^{-10}$ and $\theta = 10^{-4}$.

The results demonstrate, as could be expected that large values of ϵ, the threshold for switching to the use of grossone, imply an increased number of grossone-based steps, with corresponding lack of accuracy. In fact, it can be seen that the ratio of grossone-based steps over the total number of serious steps is smaller as ϵ decreases. Summing up, the use of grossone may allow reasonable treatment of ill-conditioning provided that the threshold ϵ is sufficiently small.

The proposed `Grossone-D-Bundle` approach has been also numerically compared against its standard counterpart, namely, Algorithm 1 where at Step 14 formula (16) for calculating H_k replaces formula (21).

The comparison is made for different values of ϵ in (15) and by stopping the algorithm after a given number N_f of function evaluations.

We report in Table 8 the results obtained by focusing on problems with size $n = 100$, and setting $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-10}\}$ and $N_f \in \{50, 100, 200\}$. In particular, we provide the relative errors $e_r^{\text{alg}} = \frac{|f^* - f(x^*)|}{1+|f(x^*)|}$, where alg $= D$ and alg $= G$ refer, respectively, to $B^k = B_{①}$ and $B^k = B_\epsilon$. It is worth noting that the results provided by the `Grossone-D-Bundle` approach are most of the times better than the standard approach, this outcome being more evident for smaller values of ϵ.

Acknowledgements

The work of M.S. Mukhametzhanov has been supported by the project No. 15-11-30022 "Global optimization, supercomputing computations, and applications" of the Russian Science Foundation.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	-66.7329331	3.65E-02	8	5	-135.7207701	3.04E-02	8	6	-276.9103190	1.60E-02	8	6
100	-67.1750430	3.02E-02	11	8	-135.9990819	2.84E-02	10	8	-277.4285342	1.42E-02	10	8
200	-68.1049532	1.69E-02	16	13	-136.1352791	2.75E-02	12	10	-277.4421523	1.41E-02	12	10
300	-68.4659193	1.18E-02	21	18	-136.8200990	2.26E-02	15	13	-277.5469159	1.37E-02	14	12
400	-68.4721827	1.17E-02	22	19	-138.7923614	8.62E-03	20	18	-277.5469159	1.37E-02	14	12
500	-68.4809312	1.16E-02	24	21	-139.3581970	4.60E-03	24	22	-277.5469159	1.37E-02	14	12

Table 2: Results on the Chained LQ test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	142.0278332	4.45E-01	7	5	276.0654964	3.92E-01	8	5	513.8191333	2.90E-01	7	5
100	118.9363346	2.11E-01	10	8	223.3313657	1.27E-01	11	8	411.7056803	3.44E-02	11	9
200	113.6568233	1.58E-01	16	14	218.4138518	1.03E-01	16	13	403.6445105	1.41E-02	16	14
300	104.2485194	6.31E-02	20	18	218.0286772	1.01E-01	20	17	400.5369972	6.36E-03	20	18
400	102.7413588	4.79E-02	25	23	200.6511044	1.33E-02	27	24	398.4287379	1.07E-03	24	22
500	99.2882901	1.30E-02	30	28	199.0866922	5.46E-03	31	28	398.1830522	4.59E-04	27	25

Table 3: Results on the Chained CB3 I test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	125.5297664	2.78E-01	7	4	227.4325378	1.48E-01	7	4	458.0950749	1.51E-01	8	5
100	109.5757036	1.17E-01	10	7	203.0976658	2.56E-02	11	8	428.6046364	7.67E-02	10	7
200	105.7547185	7.83E-02	13	10	202.4957777	2.26E-02	12	9	424.6396931	6.68E-02	12	9
300	101.5635593	3.60E-02	17	14	202.4957777	2.26E-02	12	9	424.6396931	6.68E-02	12	9
400	100.9229376	2.95E-02	22	19	202.4957777	2.26E-02	12	9	423.8254683	6.47E-02	14	11
500	100.6930428	2.72E-02	24	21	202.4957777	2.26E-02	12	9	409.6297910	2.91E-02	19	16

Table 4: Results on the Chained CB3 II test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	-69.0981172	2.82E-03	10	4	-139.4816288	3.73E-03	9	4	-280.4361446	3.51E-03	9	4
100	-69.1800716	1.66E-03	13	7	-139.7674121	1.70E-03	12	6	-280.8968967	1.88E-03	11	5
200	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
300	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
400	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
500	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5

Table 5: Results on the Chained LQ test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	100.6694475	2.70E-02	8	1	199.9535378	9.82E-03	9	3	400.6092141	6.54E-03	8	1
100	98.2901649	2.93E-03	13	3	199.9446641	9.77E-03	9	3	398.8440820	2.12E-03	10	3
200	98.2261786	2.28E-03	17	4	199.2083426	6.07E-03	14	8	398.4688649	1.18E-03	13	5
300	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.4288154	1.07E-03	17	8
400	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.2626085	6.58E-04	20	11
500	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.2282233	5.72E-04	23	14

Table 6: Results on the Chained CB3 I test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	111.4655678	1.36E-01	9	5	210.9095945	6.49E-02	9	5	440.0572743	1.05E-01	9	5
100	106.0858985	8.17E-02	11	7	201.9139633	1.97E-02	14	10	404.7495782	1.69E-02	12	8
200	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
300	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
400	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
500	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8

Table 7: Results on the Chained CB3 II test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	N_f	$\epsilon = 10^{-2}$		$\epsilon = 10^{-5}$		$\epsilon = 10^{-10}$	
		e_r^G	e_r^D	e_r^G	e_r^D	e_r^G	e_r^D
Chained LQ	50	3.04E-02	4.75E-01	7.64E-03	7.54E-01	3.73E-03	7.54E-01
	100	2.84E-02	2.13E-01	7.64E-03	7.49E-01	1.70E-03	7.54E-01
	200	2.75E-02	5.73E-02	7.64E-03	6.88E-01	1.67E-03	7.54E-01
Chained CB3-I	50	3.92E-01	7.43E-02	7.86E-03	1.05E-02	9.82E-03	1.05E-02
	100	1.27E-01	7.43E-02	5.76E-03	1.05E-02	9.77E-03	1.05E-02
	200	1.03E-01	1.36E-02	5.51E-04	1.05E-02	6.07E-03	1.05E-02
Chained CB3-II	50	1.48E-01	1.16E+00	6.47E-02	3.31E+00	6.49E-02	3.31E+00
	100	2.56E-02	5.47E-01	5.97E-02	1.42E+00	1.97E-02	2.98E+00
	200	2.26E-02	3.49E-01	5.97E-02	2.30E-01	1.69E-02	2.98E+00

Table 8: Comparisons on Chained LQ, Chained CB3 I, and Chained CB3 II, with size $n = 100$.

References

[1] P. Amodio, F. Iavernaro, F. Mazzia, M. Mukhametzhanov, Ya. D. Sergeyev, A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic, Mathematics and Computers in Simulation, doi:10.1016/j.matcom.2016.03.007, in press.

[2] A. Astorino, M. Gaudioso, E. Gorgone, A method for convex minimization based on translated first order approximations, Numerical Algorithms, doi:10.1007/s11075-017-0280-6, 2017, in press.

[3] A. M. Bagirov, N. Karmitsa, M. M. Mäkelä, Introduction to Nonsmooth Optimization: Theory, Practice and Software, Springer, 2014.

[4] A. Bagirov, B. Karasozen, M. Sezer, Discrete Gradient Method: Derivative-Free Method for Nonsmooth Optimization, Journal of Optimization Theory and Applications 137 (2008) 317–334.

[5] R. Byrd, J. Nocedal, R. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Mathematical Programming 63 (1994) 129–156.

[6] E. W. Cheney, A. A. Goldstein, Newton's method for convex programming and Tchebycheff approximation, Numerische Mathematik 1 (1959) 253–268.

[7] L. D'Alotto, Cellular Automata Using Infinite Computations, Applied Mathematics and Computation 218(16) (2012) 8077–8082.

[8] S. De Cosmis, R. De Leone, The use of Grossone in Mathematical Programming and Operations Research, Applied Mathematics and Computation 218(16) (2012) 8029–8038.

[9] A. V. Demyanov, A. Fuduli, G. Miglionico, A bundle modification strategy for convex minimization, European Journal of Operational Research 180 (2007) 38–47.

[10] V. F. Demyanov, V. N. Malozemov, Introduction to Minimax, Wiley, 1974.

[11] J. E. Dennis, Jr., J. J. Moré, Quasi-Newton Methods, Motivation and Theory, SIAM Review 19 (1977) 46–89.

[12] A. Frangioni, B. Gendron, E. Gorgone, On the Computational Efficiency of Subgradient Methods: a Case Study in Combinatorial Optimization, Tech. Rep. 2015-41, CIRRELT, Montreal, Canada, 2015.

[13] A. Fuduli, M. Gaudioso, Tuning strategy for the proximity parameter in convex minimization, Journal of Optimization Theory and Applications 130 (2006) 95–112.

[14] A. Fuduli, M. Gaudioso, G. Giallombardo, G. Miglionico, A partially inexact bundle method for convex semi-infinite minmax problems, Communications in Nonlinear Science and Numerical Simulation 21 (2014) 172–180.

[15] J.-B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms I–II, Springer-Verlag, Berlin, 1993.

[16] J. E. Kelley, The cutting-plane method for solving convex programs, Journal of SIAM 8(4) (1960) 703–712.

[17] N. Haarala, K. Miettinen, M. M. Mäkelä, Globally Convergent Limited Memory Bundle Method for Large-Scale Nonsmooth Optimization, Mathematical Programming 109(1) (2007) 181–205.

[18] J. Herskovits, E. Goulart, Sparse quasi-Newton matrices for large scale nonlinear optimization, in: Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization, 2005.

[19] D. Iudin, Ya. D. Sergeyev, M. Hayakawa, Infinity computations in cellular automaton forest-fire model, Communications in Nonlinear Science and Numerical Simulation 20(3) (2015) 861–870.

[20] N. Karmitsa, Diagonal Bundle Method for Nonsmooth Sparse Optimization, Journal of Optimization Theory and Applications 166(3) (2015) 889–905.

[21] K. C. Kiwiel, Methods of descent for nondifferentiable optimization, vol. 1133 of *Lecture notes in mathematics*, Springer-Verlag, 1985.

[22] K. C. Kiwiel, Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization, Mathematical Programming 46 (1990) 105–122.

[23] C. Lemaréchal, An algorithm for minimizing convex functions, in: J. L. Rosenfeld (Ed.), Proceedings IFIP '74 Congress 17, North-Holland, Amsterdam, 552–556, 1974.

[24] C. Lemaréchal, C. Sagastizabal, Variable Metric Bundle Methods: From Conceptual to Implementable Forms, Mathematical Programming 76 (1997) 393–410.

[25] G. Lolli, Metamathematical Investigations on the Theory of Grossone, Applied Mathematics and Computation 255 (2015) 3–14.

[26] L. Lukšan J. Vlček, A Bundle-Newton Method for Nonsmooth Unconstrained Minimization, Mathematical Programming 83 (1998) 373–391.

[27] L. Lukšan, J. Vlček, Globally Convergent Variable Metric Method for Convex Nonsmooth Unconstrained Minimization, Journal of Optimization Theory and Applications 102 (1999) 593–613.

[28] M. Margenstern, Fibonacci words, hyperbolic tilings and grossone, Communications in Nonlinear Science and Numerical Simulation 21(1–3) (2015) 3–11.

[29] F. Mazzia, Ya. D. Sergeyev, F. Iavernaro, P. Amodio, M. Mukhametzhanov, Numerical methods for solving ODEs on the Infinity Computer, in Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms", vol. 1776, AIP Publishing, New York, doi:10.1063/1.4965397, 2016.

[30] Y. Nesterov, Smooth minimization of non-smooth functions, Mathematical Programming 103 (2005) 127–152.

[31] D. Rizza, Supertasks and numeral systems, in Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms", vol. 1776, AIP Publishing, New York, doi:10.1063/1.4965369, 2016.

[32] A. Robinson, Non-standard Analysis, Princeton Univ. Press, Princeton, 1996.

[33] Ya. D. Sergeyev, Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems, EMS Surveys in Mathematical Sciences, To appear.

[34] Ya. D. Sergeyev, Numerical computations and mathematical modelling with infinite and infinitesimal numbers , Journal of Applied Mathematics and Computing 29(1) (2009) 177–195.

[35] Ya. D. Sergeyev, Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains, Nonlinear Analysis Series A: Theory, Methods & Applications 71(12) (2009) 1688–1707.

[36] Ya. D. Sergeyev, Arithmetic of Infinity, Edizioni Orizzonti Meridionali, CS, 2003, 2nd ed. 2013.

[37] Ya. D. Sergeyev, Using blinking fractals for mathematical modelling of processes of growth in biological systems, Informatica 22(4) (2011) 559–576.

[38] Ya. D. Sergeyev, M. Mukhametzhanov, F. Mazzia, F. Iavernaro, P. Amodio, Numerical methods for solving initial value problems on the Infinity Computer, International Journal of Unconventional Computing 12(1) (2016) 3–23.

[39] N. Shor, Minimization methods for nondifferentiable functions, Springer-Verlag, Berlin, 1985.

[40] P. Wolfe, A method of conjugate subgradients for minimizing nondifferentiable functions, in: M. Balinski, P. Wolfe (Eds.), Nondifferentiable optimization, vol. 3 of Mathematical Programming Study, North-Holland, Amsterdam, 145–173, 1975.

[41] A. Zhigljavsky, Computing sums of conditionally convergent and divergent series using the concept of grossone, Applied Mathematics and Computation 218(16) (2012) 8064–8076.

[42] A. Žilinskas, On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions, Applied Mathematics and Computation 218(16) (2012) 8131–8136.