The “Second Derivative” of a Non-Differentiable Function and its Use in Interval Optimization Methods

Bartłomiej Jacek Kubica

Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland

Abstract—The paper presents an idea to use weak derivatives in interval global optimization. It allows using the Newton operator to narrow domains of non-differentiable functions. Preliminary computational experiments are also presented.

Keywords—Dirac delta, distributions, generalized derivative, interval computations, interval Newton method, non-differentiable optimization.

1. Introduction

Optimization algorithms for differentiable problems are well established and sophisticated. Also for non-smooth, but Lipschitz-continuous objective functions there are well-known methods. Replacing gradients with so-called sub-gradients allows to create analogs of several gradient-based algorithms virtually no changes are needed [1]. In this paper it is proposed to extend this approach to using an analog of the second derivative.

2. Interval methods

Interval methods are a robust approach to global optimization. Here, we shall recall some basic notions of intervals and their arithmetic. We follow a widely acknowledged standard [1]:

\[ a \cdot [1/b, 1/b] \text{ for } 0 \notin b \]
\[ [-\infty, +\infty] \text{ for } 0 \in a \text{ and } 0 \in b \]
\[ \frac{a}{b}, +\infty \text{ for } \frac{a}{b} < 0 \text{ and } b < 0 = 0 \]
\[ [-\infty, \frac{a}{b}] \cup \frac{a}{b}, +\infty \text{ for } \frac{a}{b} < 0 \text{ and } b < 0 = 0 \]
\[ [-\infty, \frac{a}{b}] \cup \frac{a}{b}, +\infty \text{ for } 0 < a \text{ and } b < 0 = 0 \]
\[ [\frac{a}{b}, b, +\infty \text{ for } a < 0 \text{ and } 0 = b < b] \]
\[ \emptyset \text{ for } 0 \notin a \text{ and } 0 = b \]

The definition of interval vector \( x \), a subset of \( \mathbb{R}^n \) is straightforward: \( \mathbb{R}^n \supset x = x_1 \times \cdots \times x_n \). Traditionally interval vectors are called boxes.

Links between real and interval functions are set by the notion of an inclusion function, see, e.g., [3]; also called an interval extension, e.g., [1].

Definition 1: A function \( f : \mathbb{R}^n \to \mathbb{R} \) is an inclusion function of \( f : \mathbb{R} \to \mathbb{R} \), if for every interval \( x \) within the domain of \( f \) the following condition is satisfied:

\[ \{ f(x) \mid x \in x \} \subseteq f(\bar{x}) \]

The definition is analogous for functions \( f : \mathbb{R}^n \to \mathbb{R}^m \). When computing interval operations, we can round the lower bound downward and the upper bound upward. This will result in an interval that will be a bit overestimated, but will be guaranteed to contain the true result of the real-number operation.

Using these notions we can formulate the interval branch-and-bound (b&b) optimization algorithm, in the following way:

Branch-and-bound-method (\( x^{(0)}, f); // x^{(0)} \) is the initial box
// \( f(\cdots) \) is the interval extension of the objective function
// \( L_{sol} \) is the list of solutions
\[ y^{(0)}, \bar{y}^{(0)} = f(x^{(0)}); \]
compute \( f_{min} = \) the upper bound on the global minimum (e.g. objective value in a feasible point);
\[ L = \{ (x^{(0)}, y^{(0)}) \} \]
\[ L_{sol} = \emptyset; \]
\[ \text{while } (L \neq \emptyset) \text{ do} \]
\[ \quad x = \text{the element of } L \text{ with the lowest function value underestimation;} \]
\[ \quad \text{compute the values of interval extensions of the constraint functions;} \]
\[ \quad \text{if } (x \text{ is infeasible)} \text{ then discard } x; \]
\[ \quad \text{update } f_{\min} \text{ if possible;} \]
\[ \quad \text{perform other rejection/reduction tests on } x; \]
\[ \quad \text{if } (x \text{ is verified to contain a unique critical point or } x \text{ is small and not infeasible}) \text{ then add } x \text{ to } L_{sol}; \]
\[ \text{else} \]
\[ \quad \text{biset } x \text{ to subboxes } x^{(1)} \text{ and } x^{(2)}; \]
\[ \quad \text{compute lower bounds } y^{(1)} \text{ and } y^{(2)} \text{ on the function value in the obtained boxes;} \]
\[ \quad \text{delete } x \text{ from } L; \]
\[ \quad \text{for } i = 1, 2 \text{ do} \]
\[ \quad \quad \text{put } (x^{(i)}, y^{(i)}) \text{ on the list } L \text{ preserving the increasing order of the lower bounds;} \]
\[ \quad \text{end for} \]
\[ \quad \text{delete from } L \text{ boxes with } y^{(i)} > f_{\min}; \]
\[ \quad \text{end if} \]
\[ \text{end while} \]
\[ \text{return } L_{sol}; \]
\[ \text{end Branch-and-bound-method} \]

3. Using Weak Derivatives of a Function

What is less obvious is that for non-smooth problems we can also have an analog of the second derivative. All integrable functions – even non-differentiable ones – have so-called Sobolev generalized derivatives (also known as weak derivatives). These derivatives do not have to be functions, but may belong to a wider class of distributions (generalized functions) sometimes (see, e.g. [5], [6], [7]).

Consider the following examples:

**Example 1** – the absolute value function.

\[ f(x) = |x|, \ x \in [-3,5]. \]

Its weak derivative can be expressed as the following multifunction:

\[ \frac{Df}{Dx} = \begin{cases} -1, & \text{for } x < 0, \\ 1, & \text{for } x > 0, \\ [-1,1], & \text{for } x = 0. \end{cases} \]

More precisely, the weak derivative is a selection of the above multifunction.

And the second derivative is the following distribution:

\[ \frac{D^2 f}{Dx^2} = 2\delta(x). \]

We can approximate it using interval methods. The only thing we need is an interval extension of the Dirac delta.

And such an extension can be developed quite simply:

\[ \delta(x) = \begin{cases} [0,0], & \text{if } 0 \notin x, \\ [0, +\infty), & \text{else.} \end{cases} \] (1)

The classical Newton operator takes – in the univariate case – a well known form:

\[ x^{\text{new}} = N(x) = x - \frac{f'(x)}{f''(x)}. \]

When the second derivative is the Dirac delta, the above operator does not seem very useful. Dirac delta returns either 0 (mostly) or infinity – making the division operation undefined. However, using formula (1) and the interval Newton operator (see, e.g., [1], [2]):

\[ N(x) = \text{mid}x - \frac{f'(\text{mid}x)}{f''(x')}, \]

\[ x^{\text{new}} = N(x) \cap x, \]

we can still obtain useful results. For the interval \( x = [-3,5] \) we get \( \text{mid}x = 1, f'(\text{mid}x) = 1 \) and \( f''(x) = 0, \infty \), so – using the Kahan-Novoa-Ratz extended interval arithmetic [1] – we obtain:

\[ N(x) = 1 - \frac{1}{[0, \infty]} = 1 - [0, \infty] = [-0.5, 1], \]

which intersected with the original interval \( [-3,5] \) gives \( [-3,1] \): an interval reduced by half.

Please note that the function is not monotone in the considered interval, so no monotonicity test (using only subgradients) would allow us to narrow the domain without branching.

The above result has been obtained for the least promising case probably. Let us consider a “more friendly” objective.

**Example 2.**

\[ f(x) = x^2 + |x|, \ x \in [-3,5]. \]

Its weak derivative can be expressed as:

\[ \frac{Df}{Dx} = \begin{cases} 2x - 1, & \text{for } x < 0, \\ 2x + 1, & \text{for } x > 0, \\ [-1,1], & \text{for } x = 0. \end{cases} \]

The second derivative is:

\[ \frac{D^2 f}{Dx^2} = 2 + 2\delta(x). \]

Now for the interval \( x = [-3,5] \), we get:

\[ N(x) = 1 - \frac{3}{[2, \infty]} = 1 - [0, 1.5] = [-0.5, 1]. \]

So, both endpoints of the interval have been improved! The remainder of the paper considers examples of problems that can be solved using the proposed methodology.
Apparently, use of the second weak derivative is superior to using generalized gradients only — at least for some problems.

4. Computational Experiments

Numerical experiments were performed on a computer with 16 cores, i.e., 8 Dual-Core AMD Opterons 8218 with 2.6 GHz clock. The machine ran under control of a Fedora 15 Linux operating system. The solver was implemented in C++, using C-XSC 2.5.1 library [8] for interval computations and automatic differentiation. To deal with non-smooth functions, e.g., min() and max(), the automatic differentiation code had to be modified (files hess_ari.hpp and hess_ari.cpp). The interval global optimization algorithm, on the other hand, did not have to be modified significantly (see e.g. [1]); only some maintenance changes were done.

The following optimization problems were considered.

\[
\min_{x} f_1(x) = \sum_{i=1}^{n} |x_i| ,
\]

\[\text{s.t.} \quad x_i \in [-3,5.5], \quad i = 1, \ldots, n .\]

\[
\min_{x} f_2(x) = \sum_{i=1}^{n} (x_i^2 + |x_i|) ,
\]

\[\text{s.t.} \quad x_i \in [-3,5.5], \quad i = 1, \ldots, n .\]

\[
\min_{x} f_3(x) = \sum_{i=1}^{n} (-1)^{i+1} \cdot |x_i| ,
\]

\[\text{s.t.} \quad x_i \in [-3,5.5], \quad i = 1, \ldots, n .\]

\[
\min_{x_1,x_2} f_4(x_1,x_2,x_3) = |x_1^2 - 1| + |x_2| - |x_3| ,
\]

\[\text{s.t.} \quad x_1,x_2,x_3 \in [-3,5.5] .\]

\[
\min_{x_1,x_2} f_5(x) = \max\{-(x-1)^2, -(x+1)^2\} ,
\]

\[\text{s.t.} \quad x \in [-0.5,1.0] .\]

Tables 1–5 contain the results for these problems. Following fields are:

- **var** – the number of decision variables of the problem (for functions (2)–(6), where this number may be arbitrary),
- **fun** – the number of objectives evaluations required,
- **grad** – the number of objective’ gradients evaluations required.
- **Hesse** – the number of objective’ Hesse matrices evaluations required,
- **bis** – the number of boxes’ bisections required,
- **box** – the number of resulting boxes, approximating the set of solutions.

Fields “fun”, “grad”, “Hesse” and “bis” are all some kind of measure of the algorithm performance. It seems the number of bisections (roughly corresponding to the number of iterations) is the best measure, but all of them should be taken into account.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Results for problem Eq. (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>fun</td>
</tr>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>387</td>
</tr>
<tr>
<td>16</td>
<td>771</td>
</tr>
<tr>
<td>32</td>
<td>1539</td>
</tr>
<tr>
<td>64</td>
<td>3075</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Results for problem Eq. (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>fun</td>
</tr>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>387</td>
</tr>
<tr>
<td>16</td>
<td>771</td>
</tr>
<tr>
<td>32</td>
<td>1539</td>
</tr>
<tr>
<td>64</td>
<td>3075</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Results for problem Eq. (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>fun</td>
</tr>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>387</td>
</tr>
<tr>
<td>16</td>
<td>771</td>
</tr>
<tr>
<td>32</td>
<td>1539</td>
</tr>
<tr>
<td>64</td>
<td>3075</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Results for problem Eq. (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>fun</td>
</tr>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>387</td>
</tr>
<tr>
<td>16</td>
<td>771</td>
</tr>
<tr>
<td>32</td>
<td>1539</td>
</tr>
<tr>
<td>64</td>
<td>3075</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Results for problem Eq. (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>fun</td>
</tr>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>387</td>
</tr>
<tr>
<td>16</td>
<td>771</td>
</tr>
<tr>
<td>32</td>
<td>1539</td>
</tr>
<tr>
<td>64</td>
<td>3075</td>
</tr>
</tbody>
</table>
5. Results

In all cases the proposed method performed much better than the one not using the second-order information (in terms of number of bisections, Hesse matrices evaluation, etc.). For two variables, using the 2nd weak derivatives seems to reduce the number of iterations by the factor of 2. For higher dimensions, it becomes much larger – for 64 variables it is even 10 times faster.

An interesting result was obtained for problem Eq. (3), in Table 2. The function is convex and the proposed version of the Newton operator allows – as shown in Example 2 – narrowing the interval on both sides. Thus, the number of bisections required is equal to the problem dimension multiplied by 2; it is sufficient to cut off the bounds and the interior can be narrowed instantly.

Yet more interesting results can be found in Table 4. Function is nonconvex, yet the performance of the b&b algorithm seems even better than for convex functions. Also, speedups gained by the use of 2nd weak derivatives seem as good as for convex problems. This – very desired – phenomenon should be investigated in subsequent papers, deeper.

6. Conclusions and future work

Results presented in this paper are only preliminary, yet very promising. For several simple functions the algorithm using generalized second order derivatives required far fewer iterations than the one using 1st order subderivatives only.

Some theoretical investigation of the convergence is required. It does not seem that using the Newton operator used with weak derivatives for non-differentiable functions results in quadratic order of convergence; yet it is more efficient than not using this tool, the monotonicity test only. The specific order of convergence is to be determined, yet. Finally, the presented approach is an interesting example of the difference of features of interval and non-interval algorithms – the non-interval version of the Newton operator based on pseudo-functions seems completely inapplicable, as discussed in Section 3.

Acknowledgments

The paper has been done as a part of realization of the grant for statutory activity, financed by the Dean of Faculty of Electronics and Information Technology (WUT), titled “Interval methods for solving nonlinear problems”.

References

Bartłomiej J. Kubica received his Ph.D. in Computer Science in 2006 from the Warsaw University of Technology. Since 2005 with WUT. Currently an assistant professor in Complex Systems Group. He co-organizes interval sessions at PPAM conferences and organizes at PARA. He co-authored a book on parallel programming and wrote several papers and presentations. His research interests focus on interval methods, optimization algorithms, multicriteria decision making, game theory and – on the other hand – multithreaded programming and parallel computations. E-mail: bkubica@elka.pw.edu.pl
Institute of Control and Computation Engineering Warsaw University of Technology
Nowowiejska 15/19
00-665 Warsaw, Poland