Paper

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 subgradients allows to create analogs of several gradient-based methods for non-differentiable problems. For interval 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 standards (cf., e.g., [2], [3], [1]).
We define the (closed) interval $[\underline{x},\overline{x}]$ as a set $\{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$.
Following [4], we use boldface lowercase letters to denote interval variables, e.g., $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and $\mathbb{IR}$ denotes the set of all real intervals.
We design arithmetic operations on intervals so that the following condition is fulfilled: if we have $\odot \in \{+,-,\cdot,/\}$, $a \in \mathbf{a}$, $b \in \mathbf{b}$, then $a \odot b \in \mathbf{a} \odot \mathbf{b}$. The actual formulae for arithmetic operations (see, e.g., [1], [2]) are as follows:

$$
\begin{aligned}
[\underline{a},\overline{a}] + [\underline{b},\overline{b}] &= [\underline{a}+\underline{b},\overline{a}+\overline{b}] \ , \\
[\underline{a},\overline{a}] - [\underline{b},\overline{b}] &= [\underline{a}-\overline{b},\overline{a}-\underline{b}] \ , \\
[\underline{a},\overline{a}] \cdot [\underline{b},\overline{b}] &= [\min(\underline{ab},\underline{a}\overline{b},\overline{a}\underline{b},\overline{ab}),\max(\underline{ab},\underline{a}\overline{b},\overline{a}\underline{b},\overline{ab})] \ , \\
[\underline{a},\overline{a}] \ / \ [\underline{b},\overline{b}] &= [\underline{a},\overline{a}] \cdot \left[1/\overline{b},1/\underline{b}\right] \ , \qquad 0 \notin [\underline{b},\overline{b}] \ .
\end{aligned}
$$

The so-called *extended interval arithmetic* allows division by an interval containing zero, not covered by the above formulae. The basic idea is that the result of such a division should be the set of all possible results of the division operation, executed on numbers from the argument intervals. We give here the formulae of Kahan–Novoa–Ratz arithmetic, following [1]:

$$
\mathbf{a}/\mathbf{b} = \begin{cases}
\mathbf{a} \cdot [1/\overline{b},1/\underline{b}] & \text{for } 0 \notin \mathbf{b} \\
[-\infty,+\infty] & \text{for } 0 \in \mathbf{a} \text{ and } 0 \in \mathbf{b} \\
[\overline{a}/\underline{b},+\infty] & \text{for } \overline{a} < 0 \text{ and } \underline{b} < \overline{b} = 0 \\
[-\infty,\overline{a}/\overline{b}] \cup [\overline{a}/\underline{b},+\infty] & \text{for } \overline{a} < 0 \text{ and } \underline{b} < 0 < \overline{b} \\
[-\infty,\overline{a}/\overline{b}] & \text{for } \overline{a} < 0 \text{ and } 0 = \underline{b} < \overline{b} \\
[-\infty,\underline{a}/\underline{b}] & \text{for } 0 < \underline{a} \text{ and } \underline{b} < \overline{b} = 0 \\
[-\infty,\underline{a}/\underline{b}] \cup [\underline{a}/\overline{b},+\infty] & \text{for } 0 < \underline{a} \text{ and } \underline{b} < 0 < \overline{b} \\
[\underline{a}/\overline{b},+\infty] & \text{for } \underline{a} < 0 \text{ and } 0 = \underline{b} < \overline{b} \\
\emptyset & \text{for } 0 \notin \mathbf{a} \text{ and } 0 = \mathbf{b}
\end{cases} .
$$

The definition of interval vector $\mathbf{x}$, a subset of $\mathbb{R}^n$ is straightforward: $\mathbb{R}^n \supset \mathbf{x} = \mathbf{x}_1 \times \cdots \times \mathbf{x}_n$. 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 $\mathsf{f}\colon \mathbb{IR} \to \mathbb{IR}$ is an *inclusion function* of $f\colon \mathbb{R} \to \mathbb{R}$, if for every interval $\mathbf{x}$ within the domain of f the following condition is satisfied:

$$\{f(x) \mid x \in \mathbf{x}\} \subseteq \mathsf{f}(\mathbf{x}) \ .$$

The definition is analogous for functions $f\colon \mathbb{R}^n \to \mathbb{R}^m$.
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` $(\mathbf{x}^{(0)}, \mathsf{f})$;
// $\mathbf{x}^{(0)}$ is the initial box
// $\mathsf{f}(\cdots)$ is the interval extension of the objective function
// L_{sol} is the list of solutions
$[\underline{y}^{(0)}, \overline{y}^{(0)}] = \mathsf{f}(\mathbf{x}^{(0)})$;
compute f_{min} = the upper bound on the global minimum
(e.g. objective value in a feasible point);
$L = \{(\mathbf{x}^{(0)}, \underline{y}^{(0)})\}$;

$L_{sol} = \emptyset$;
`while` ($L \neq \emptyset$) `do`
    x = the element of L with the lowest function value underestimation;
    compute the values of interval extensions of the constraint functions;
    `if` (**x** is infeasible) `then` discard **x**;
    update f_{min} if possible;
    perform other rejection/reduction tests on **x**;
    `if` (**x** is verified to contain a unique critical point or **x** is small and not infeasible) `then`
        add **x** to L_{sol};
    `else`
        bisect **x** to subboxes $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$;
        compute lower bounds $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ on the function value in the obtained boxes;
        delete **x** from L;
        `for` $i = 1, 2$ `do`
            put $(\mathbf{x}^{(i)}, \underline{y}^{(i)})$ on the list L preserving the increasing order of the lower bounds;
        `end for`
        delete from L boxes with $\underline{y}^{(i)} > f_{min}$;
    `end if`
`end while`
delete from L_{sol} the boxes with $\underline{y}^{(i)} > f_{min}$;
`return` L_{sol};
`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:

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

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

$$x^{new} = N(x) = x - \frac{f'(x)}{f''(x)} \ .$$

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]):

$$\begin{aligned} N(\mathbf{x}) &= \operatorname{mid}\mathbf{x} - \frac{\mathsf{f}'(\operatorname{mid}\mathbf{x})}{\mathsf{f}''(x)} \ , \\ \mathbf{x}^{new} &= N(\mathbf{x}) \cap \mathbf{x} \ , \end{aligned}$$

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

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

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.

$$\min_x f_1(x) = \sum_{i=1}^{n} |x_i| \ , \tag{2}$$
s.t.
$$x_i \in [-3, 5.5], \ i = 1, \ldots, n \ .$$

$$\min_x f_2(x) = \sum_{i=1}^{n} \left(x_i^2 + |x_i|\right) \ , \tag{3}$$
s.t.
$$x_i \in [-3, 5.5], \ i = 1, \ldots, n \ .$$

$$\min_x f_3(x) = \sum_{i=1}^{n} (-1)^{i+1} \cdot |x_i| \ , \tag{4}$$
s.t.
$$x_i \in [-3, 5.5], \ i = 1, \ldots, n \ .$$

$$\min_{x_1, x_2} f_4(x_1, x_2, x_3) = |x_1^4 - 1| + |x_2| - |x_3| \ , \tag{5}$$
s.t.
$$x_1, x_2, x_3 \in [-3, 5.5] \ .$$

$$\min_{x_1, x_2} f_5(x) = \max\{-(x-1)^2, -(x+1)^2\} \ , \tag{6}$$
s.t.
$$x \in [-0.5, 1.0] \ .$$

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 1
Results for problem Eq. (2)

	1st order				
var	fun	grad	Hesse	bis	box
1	51	27	25	24	1
2	99	53	49	48	1
4	195	113	97	96	1
8	387	225	193	192	1
16	771	481	385	384	1
32	1539	961	769	768	1
64	3075	2049	1537	1536	1
	2nd order				
var	fun	grad	Hesse	bis	box
1	29	26	14	13	1
2	39	35	19	18	1
4	53	46	26	25	1
8	73	59	36	35	1
16	107	79	53	52	1
32	173	114	86	85	1
64	301	179	150	149	1

Table 2
Results for problem Eq. (3)

	1st order				
var	fun	grad	Hesse	bis	box
1	51	27	25	24	1
2	99	53	49	48	1
4	195	113	97	96	1
8	387	225	193	192	1
16	771	481	385	384	1
32	1539	961	769	768	1
64	3075	2049	1537	1536	1
	2nd order				
var	fun	grad	Hesse	bis	box
1	7	24	23	2	1
2	11	26	25	4	1
4	19	30	29	8	1
8	35	38	37	16	1
16	67	54	53	32	1
32	131	86	85	64	1
64	259	150	149	128	1

Table 3
Results for problem Eq. (4)

	1st order				
var	fun	grad	Hesse	bis	box
1	51	27	25	24	1
2	53	28	26	25	1
4	103	55	51	50	1
8	203	117	101	100	1
16	403	233	201	200	1
32	803	497	401	400	1
64	1603	993	801	800	1
	2nd order				
var	fun	grad	Hesse	bis	box
1	29	26	14	13	1
2	37	21	15	14	1
4	43	37	21	20	1
8	61	50	30	29	1
16	89	67	44	43	1
32	139	95	69	68	1
64	237	146	118	117	1

Table 4
Results for problem Eq. (5)

	fun	grad	Hesse	bis	box
1st order	192	107	96	94	2
2nd order	88	83	44	42	2

Table 5
Results for problem Eq. (6)

	fun	grad	Hesse	bis	box
1st order	45	24	22	21	1
2nd order	25	23	12	11	1

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

[1] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*. Dordrecht: Kluwer, 1996.

[2] E. Hansen and W. Walster, *Global Optimization Using Interval Analysis*. New York: Marcel Dekker, 2004.

[3] L. Jaulin, M. Kieffer, O. Didrit and E. Walter, *Applied Interval Analysis*. London: Springer, 2001.

[4] R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary and P. van Hentenryck, “Standardized notation in interval analysis” [Online]. Available: http://www.mat.univie.ac.at/∼neum/software/int/notation.ps.gz

[5] M. J. Lighthill, *Introduction to Fourier Analysis and Generalised Functions*. Cambridge University Press, 1959.

[6] H. Marcinkowska, *Dystrybucje i Przestrzenie Sobolewa*. Wrocław: Wydawnictwo Uniwersytetu Wrocławskiego, 1990 (in Polish).

[7] V. S. Vladimirov, *Methods of the Theory of Generalized Functions*. Taylor & Francis, 2002.

[8] C-XSC library [Online]. Available: http://www.xsc.de

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