Gaussian and Cauchy Functions in the Filled Function Method – Why and What Next: On the Example of Optimizing Road Tolls

José Guadalupe Flores Muñiz[1], Vyacheslav V. Kalashnikov[2,3],
Nataliya Kalashnykova[1,4], and Vladik Kreinovich[5]
[1]Department of Physics and Mathematics
Universidad Autoónoma de Nuevo León
San Nicolás de los Garza, México
jose_guadalupe64@hotmail.com
nkalash2009@gmail.com
[2]Department of Systems and Industrial Engineering
Tecnológico de Monterrey ITESM, Campus Monterrey
Monterrey, Mexcio, kalash@itsem.mx
[3]Department of Experimental Economics
Central Economics and Mathematics Institute (CEMI)
Moscow, Russian Federation
[4]Department of Computer Science
Sumy State University
Sumy, Ukraine
[5]Department of Computer Science
University of Texas at El Paso
El Paso, Texas 79968, USA, vladik@utep.edu

Abstract

In many practical problems, we need to find the values of the parameters that optimize the desired objective function. For example, for the toll roads, it is important to set the toll values that lead to the fastest return on investment.

There exist many optimization algorithms, the problem is that these algorithms often end up in a local optimum. One of the promising methods to avoid the local optima is the filled function method, in which we, in effect, first optimize a smoothed version of the objective function, and then use the resulting optimum to look for the optimum of the original function. It turns out that empirically, the best smoothing functions to use in this method are the Gaussian and the Cauchy functions. In this paper, we show that from the viewpoint of computational complexity,

these two smoothing functions are indeed the simplest.

The Gaussian and Cauchy functions are not a panacea: in some cases, they still leave us with a local optimum. In this paper, we use the computational complexity analysis to describe the next-simplest smoothing functions which are worth trying in such situations.

1 Optimizing Road Tolls: A Brief Introduction to the Case Study

In many practical problems, we need to optimize an appropriate objective function. In this paper, as a case study, we consider the problem of optimizing road tolls; see [3] for details.

The need for road tolls comes from the fact that in many geographic locations, traffic is congested, there is a need to build new roads that would decrease this congestion. Often, however, the corresponding governments do not have the funds to build the new roads.

A solution is to build the *toll roads*, i.e., to request that the drivers pay for driving on these roads – and thus, to get back the money that was spent on building these roads. Sometimes, the governments borrow the money to build the roads, and use the collected tolls to pay back the loan. In other cases, a private company is selected to build the road: the company invests the money, and get its investment back from the collected tolls.

In both arrangements, for a system of toll roads, it is important to select the toll values that will lead to the fastest possible return on investment. This is a complex problem:

- if the tolls are too small, it will take forever to get back the investments;
- on the other hand, if the tolls are too large, then most drivers will prefer to use the existing toll-free roads, and again, it will take a long time to get back the investment.

It is therefore important to find the optimal toll values that minimize the amount of time needed to return the investment.

2 Optimization in General: How to Avoid Local Optima?

Local optima: a problem. There exist many optimization algorithms. However, often, they lead to a local optimum.

How to avoid local optima: the filled function method. One of the promising methods to avoid the local optima is the *filled function* method, in which we, in effect, first optimize a smoothed version of the objective function, and then use the resulting optimum to look for the optimum of the original function. This method was originally proposed in [5]; see also [1, 3, 6, 7].

Specifically, once we reach a local optimum x^*, then we optimize an auxiliary expression

$$K\left(\frac{x - x^*}{\sigma}\right) \cdot F(f(x), f(x^*), x) + G(f(x), f(x^*), x),$$

for appropriate functions $K(x)$, $F(f, f^*, x)$, and $G(f, f^*, x)$, and for an appropriate value σ. Once we find the optimum of this auxiliary expression, we use it as a new first approximation to find the optimum of the original objective function $f(x)$.

Filled function method: results and open problems. How well we can avoid the local optimum depends on the choice of the smoothing function $K(x)$. In [6], it was shown that for several optimization problem, the best choice is to use the Cauchy smoothing function

$$K(x) = \frac{1}{1 + \|x\|^2}.$$

For toll optimization and for several similar problems, it turned out that the Gaussian smoothing function $K(x) = \exp(-\|x\|^2)$ leads to the best results; see, e.g., [3].

In some cases, none of the known smoothing functions worked well. So, we arrive at the following natural problems:

- why are the Gaussian and Cauchy smoothing functions empirically the best?
- which smoothing function should we choose if neither Gaussian nor Cauchy smoothing functions work well?

What we do in this paper. In this paper, we provide answers to both questions.

3 Computational Complexity as a Natural Criterion for Selecting a Smoothing Function

Why computational complexity. What criterion should we use to select a smoothing function? We can always avoid a local optimum if we repeatedly start the same optimization process at several randomly selected points: if we start at many such points, one of them will be close to the global optimum. However, this will drastically increase the computation time.

The main advantage of the filled function method is that it allows us to decrease the computation time. From this viewpoint, the less time we need to compute the smoothing function, the better.

Each computation consists of several elementary computational steps, and the computation time is thus proportional to the number of such steps – maybe taken with weights. This (weighted) number of steps is known as *computational complexity*; see, e.g., [2, 4]. From this viewpoint, we want a smoothing function which has the smallest possible computational complexity.

How can we measure computational complexity. Most programming languages use the following elementary computational operations:

- arithmetic operations: unary minus ($-x$), addition, subtraction, multiplication, and division, and
- elementary functions: $\exp(x)$, $\ln(x)$, $\sin(x)$, $\cos(x)$, $\tan(x)$, $\arcsin(x)$, $\arccos(x)$, and $\arctan(x)$.

Thus, first, we need to minimize the overall number of such computational steps.

Not all these steps require the same computation time:

- unary minus ($-x$) is the fastest operation, it requires that we only change one bit: the bit describing the sign;
- addition and subtraction are next in complexity;
- multiplication takes somewhat longer, since multiplication, in effect, means several additions;
- finally, computation of elementary functions requires even longer time, since each such computation requires several multiplications and additions.

We will take this difference into account when deciding which smoothing function is the fastest to compute.

4 Analysis of the Problem and the Main Result

Natural requirements on a smoothing function. The smoothing function should be symmetric, since we have no reason to prefer different orientation of coordinates. Thus, it should depend only on $v \stackrel{\text{def}}{=} \|x\|^2$: $K(x) = g(v)$ for some function $g(v)$.

This function $g(v)$ should be finite and non-negative for all $v \geq 0$, and it should tend to 0 when $v \to +\infty$.

It is easy to see that both Gaussian and Cauchy smoothing functions satisfy these requirements, correspondingly with $g(v) = \exp(-v)$ and $g(v) = \dfrac{1}{1+v}$.

Computational complexity of the Gaussian and Cauchy smoothing functions. The function $g(v) = \exp(-v)$ (corresponding to Gaussian smoothing) requires two operations to compute:

- a unary minus, to compute $-v$, and
- the exponential function, to transform $-v$ into $\exp(-v)$.

Similarly, the function $g(v)=\dfrac{1}{1+v}$ (corresponding to Cauchy smoothing) consists of two operations:

- addition, to compute $1+v$, and
- division, to transform $1+v$ into $g(v)$.

Our first result. Our first result is a classification of all smoothing functions that can be computed in two or fewer computational steps.

Definition 1. *By a* smoothing function*, we mean a non-zero non-negative function $g(v)$ which is defined for all $v\ge 0$ and which tends to 0 as $v\to+\infty$.*

Definition 2.

- *We say that a function $g(v)$ is* computable in 0 steps *if it is either an identity $g(v)=v$ or a constant $g(v)=\mathrm{const}$.*
- *By an* elementary operation*, we mean either an arithmetic operation (unary minus, addition, subtraction, multiplication, or division), or an elementary function (*$\exp(x)$*,* $\ln(x)$*,* $\sin(x)$*,* $\cos(x)$*,* $\tan(x)$*,* $\arcsin(x)$*,* $\arccos(x)$*, or* $\arctan(x)$*).*
- *If $F(x)$ is an elementary operation, and $h(v)$ is computable in k steps, then we say that the function $g(v)=F(h(v))$ is* computable in $k+1$ steps.
- *If $F(x,y)$ is an elementary operation, and the functions $h(v)$ and $h'(v)$ are computable, correspondingly, in k and k' steps, then we say that the function $g(v)=F(h(v),h'(v))$ is* computable in $k+k'+1$ steps.

Proposition. *A smoothing function is computable in 2 steps if and only if it has one of the following forms:*

- $g(v)=\dfrac{c'}{c+v}$*, for some constants c and c',*
- $g(v)=\mathrm{const}\cdot\exp(-c\cdot v)$*,*
- $g(v)=\dfrac{\pi}{2}-\arctan(v)$*,*
- $g(v)=\arctan\left(\dfrac{1}{v}\right)$*, or*
- $g(v)=\cos(\arctan(v))$*.*

Comment. For convenience, the proof of this Proposition is given in the next section.

Among these five, which are the fastest to compute? Which of the above five functions is the fastest to compute?

1. The function $g(v) = \dfrac{1}{1+v}$ requires one addition and one multiplication.
2. The function $g(v) = \exp(-v)$ requires one unary minus and one application of an elementary function.
3. The function $g(v) = \dfrac{\pi}{2} - \arctan(v)$ requires one subtraction and one application of an elementary function.
4. The function $g(v) = \arctan\left(\dfrac{1}{v}\right)$ requires one division and one application of an elementary function.
5. Finally, the function $g(v) = \cos(\arctan(v))$ requires two applications of elementary functions.

We can now make the following comparisons:

- Since multiplication/division is faster than an application of an elementary function, and addition is faster than multiplication/division and than elementary functions, the function 1 is faster to compute than functions 3, 4, and 5.
- Similarly, since the unary minus is faster than any other operation, function 2 is faster to compute than functions 3, 4, and 5.
- Since subtraction is faster than division, function 3 is faster than function 4.
- Finally, since multiplication/division is faster than an application of an elementary function, function 4 is faster than function 5.

Thus, we arrive at the following conclusion.

Conclusion. Among all smoothing functions that can be computed in two computational steps:

- the functions $g(v) = \dfrac{1}{1+v}$ and $g(v) = \exp(-v)$ corresponding to Cauchy and Gaussian smoothing are the fastest to compute;
- next fastest is the function $g(v) = \dfrac{\pi}{2} - \arctan(v)$;
- next fastest is the function $g(v) = \arctan\left(\dfrac{1}{v}\right)$; and
- finally, the slowest to compute is the function $g(v) = \cos(\arctan(v))$.

This explains why the Gaussian and Cauchy functions are indeed empirically the best, and this also show what to do when these smoothing functions do not work well: try smoothing functions $K(x) = g(\|x\|^2)$ corresponding to $g(v) = \dfrac{\pi}{2} - \arctan(v)$, $g(v) = \arctan\left(\dfrac{1}{v}\right)$, and $g(v) = \cos(\arctan(v))$.

5 Proof of the Main Result

1°. Clearly, functions $g(v) = v$ and $g(v) = \text{const}$ which are computable in 0 steps are not smoothing functions, since they do not tend to 0 when $v \to +\infty$.

Let us show that similarly, no smoothing function can be computed in 1 step. Indeed, we can easily list all functions computable in 1 step:

$$g(v) = v + c, \quad g(v) = v - c, \quad g(v) = c - v, \quad g(v) = c \cdot v, \quad g(v) = \frac{c}{v},$$

$$g(v) = \frac{v}{c}, \quad g(v) = \exp(v), \quad g(v) = \ln(v), \quad g(v) = \sin(v), \quad g(v) = \cos(v),$$

$$g(v) = \tan(v), \quad g(v) = \arcsin(v), \quad g(v) = \arccos(v), \quad g(v) = \arctan(v),$$

where c is a constant.

From the above functions, the function $g(v) = \frac{c}{v}$ is not a smoothing function since it is not defined for $v = 0$, and all other functions are not smoothing functions since they do not satisfy the condition that $\lim\limits_{v \to +\infty} g(v) = 0$.

Thus, a smoothing function must have at least two computational steps.

2°. By definition, a function computable in 2 steps has the form $F(h(v))$, where $h(v)$ is computable in 1 step, or the form $F(h(v), h'(v))$, where $h(v)$ is computable in one step and $h'(v)$ is computable in 0 steps (i.e., is either an identity or a constant).

We have already listed all possible functions $h(v)$ which can be computed in one step. Let us consider these functions one by one.

3°. If $h(v) = v + c$, then, $h(+\infty) = +\infty$. So, for the composition to be a smoothing function, we must have $F(+\infty) = 0$.

As we showed in the 1-step case, the only operation that satisfies this condition is $g(w) = \frac{c'}{w}$, so we get $g(v) = \frac{c'}{v + c}$. This case corresponds to the Cauchy function.

4°. The case $h(v) = v - c$ is equivalent to $h(v) = v + (-c)$, so it is the same case that we have already considered.

5°. If $h(v) = c_1 - v$, then, $h(+\infty) = -\infty$, so the function $F(w)$ must satisfy the condition $F(-\infty) = 0$.

Two operations satisfy this condition: $F(w) = \frac{c'}{w}$ and $F(w) = \exp(w)$.

- If $F(w) = c'w$, then, $g(v) = \frac{c'}{c - v}$. This is equal to $g(v) = \frac{(-c')}{v + (-c)}$, i.e., to the Cauchy case that we have already considered.
- If $F(w) = \exp(w)$, then, $g(v) = \exp(c - v)$, i.e., $g(v) = \text{const} \cdot \exp(-v)$, where $\text{const} = \exp(c)$. This case corresponds to the Gaussian smoothing.

6°. If $h(v) = c \cdot v$, then, depending on the sign of c, we have different asymptotic behaviors for $h(v)$.

If $c > 0$, then we have $h(+\infty) = +\infty$. In this case, the only possibility to get $g(h) \to 0$ as $h \to +\infty$ is to have $F(w) = \frac{c'}{w}$, but in this case $g(v) = \frac{c'}{c \cdot v}$ is not defined for $v = 0$.

If $c < 0$, then $h(+\infty) = -\infty$. In this case, we similarly cannot have $F(w) = \frac{c'}{w}$, but now we have a second option $F(w) = \exp(w)$, in which case $g(v) = \exp(c \cdot v)$. This case corresponds to the Gaussian function.

7°. If $h(v) = \frac{c}{v}$, then, $h(+\infty) = 0$, so the function $F(w)$ must satisfy the condition $F(0) = 0$. Six operations satisfy this condition: $F(w) = c' \cdot w$, $F(w) = \frac{w}{c}$, $F(w) = \sin(w)$, $F(w) = \tan(w)$, $F(w) = \arcsin(w)$ and $F(w) = \arctan(w)$.

When $c > 0$, then $h(0) = +\infty$, so, additionally, $F(+\infty)$ must be finite and non-negative. This condition is satisfy only by $F(w) = \arctan(w)$, so we get $g(v) = \arctan\left(\frac{c}{v}\right)$.

When $c < 0$, then $h(0) = -\infty$, so, additionally, $F(-\infty)$ must be finite and non-negative. This condition is not met by any of the above functions $F(w)$.

8°. The case $h(v) = \frac{v}{c}$ is equivalent to the already analyzed case $h(v) = v \cdot \text{const}$, with $\text{const} = \frac{1}{c}$.

9°. If $h(v) = \exp(v)$, then, $h(+\infty) = +\infty$, so we must have $F(w) = c'w$ and $g(v) = \frac{c'}{\exp(v)}$. This is equal to $g(v) = \text{const} \cdot \exp(-v)$, i.e., corresponds to the Gaussian case.

10°. If $h(v) = \ln(v)$, then, $h(+\infty) = +\infty$, so we must have $F(w) = \frac{c'}{w}$ and, thus, $g(v) = \frac{c'}{\ln(v)}$. This function is not defined (is infinite) when $v = 1$ and thus, is not a smoothing function.

11°. If $h(v) = \sin(v)$ or $h(v) = \cos(v)$, then $h(v)$ oscillates between -1 and 1 and has no limit when $v \to +\infty$. So, for $g(v) \to 0$, the function $F(w)$ must be equal to 0 for all the values $w \in [-1, 1]$, but no elementary operation has this property.

Similarly, it is not possible to have $h(v) = \tan(v)$.

12°. If $h(v) = \arcsin(v)$ or $h(v) = \arccos(v)$, then $g(v) = F(h(v))$ cannot be a smoothing function since $h(v)$ is not defined for $v > 1$.

13°. If $h(v) = \arctan(v)$, then, $h(+\infty) = \pi/2$, so the function $F(w)$ must satisfy the condition $F\left(\frac{\pi}{2}\right) = 0$. Three elementary functions satisfy this condition: $F(w) = w - \frac{\pi}{2}$, $F(w) = \frac{\pi}{2} - w$, and $F(w) = \cos(w)$.

When $F(w) = w - \frac{\pi}{2}$, then the function $g(v) = \arctan(v) - \frac{\pi}{2}$ has negative values, so it cannot be a smoothing function.

The other two cases correspond to the last two function in the formulation of the Proposition.

The Proposition is thus proven.

Acknowledgments

This work was supported by a grant from Mexico Consejo Nacional de Ciencia y Tecnología (CONACYT). It was also partly supported by the US National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, and by an award "UTEP and Prudential Actuarial Science Academy and Pipeline Initiative" from Prudential Foundation.

This work was performed when José Guadalupe Flores Muñiz visited the University of Texas at El Paso.

References

[1] B. Addis, M. Locatelli, and F. Schoen, "Local optima smoothing for global optimization", *Optimization Methods and Software*, 2005, Vol. 20, No. 4–5, pp. 417–437.

[2] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.

[3] V. V. Kalashnikov, R. C. Herrera Maldonado, and J.-F. Camacho-Vallejo, "A heuristic algorithm solving bilevel toll optimization problem", *The International Journal of Logistics Management*, 2016, Vol. 27, No. 1, pp. 31–51.

[4] C. Papadimitriou, *Computational Complexity*, Addison Welsey, Reading, Massachusetts, 1994.

[5] G. E. Renpu, "A filled function method for finding a global minimizer of a function of several variables", *Mathematical Programming*, 1988, Vol. 46, No. 1, pp. 57–67.

[6] Z. Y. Wu, F. S. Bai, Y. J. Yang, and M. Mammadov, "A new auxiliary function method for general constrained global optimization", *Optimization*, 2013, Vol. 62, No. 2, pp. 193–210.

[7] Z. Y. Wu, M. Mammadov, F. S. Bai, and Y. J. Yang, "A filled function method for nonlinear equations", *Applied Mathematics and Computation*, 2007, Vol. 189, No. 2, pp. 1196–1204.